Universidad de Valladolid

Universidad de Valladolid
Facultad de Ciencias

Trabajo de Fin de Grado
Grado en Estadística

Evaluación de métodos de *machine learning* en la detección de genes circadianos

Autor: Aníbal Hernando Novo
Tutoras: Yolanda Larriba González, Itziar Fernández Martínez

Curso 2024–2025

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento a mis tutoras, Yolanda Larriba González e Itziar Fernández Martínez, por su constante apoyo, dedicación y orientación a lo largo del desarrollo de este Trabajo de Fin de Grado.

A mis compañeros, especialmente a aquellos con los que he compartido los momentos más intensos de este camino, por su ayuda incondicional, su compañerismo y por enseñarme tanto dentro como fuera del aula.

Y, por último, pero lo más importante, a mi madre y novia, por su apoyo inquebrantable, por estar siempre a mi lado y por motivarme a seguir creciendo tanto en lo profesional como en lo personal.

Resumen

Los ritmos circadianos en los seres vivos están controlados por un sistema interno de sincronización que regula la expresión génica en ciclos de aproximadamente 24 horas. Estos ritmos son fundamentales para numerosos procesos fisiológicos, lo que otorga a su estudio una gran relevancia en el ámbito biomédico actual. No obstante, el análisis de genes circadianos en humanos presenta importantes desafíos. Uno de los principales desafíos radica en la imposibilidad de obtener muestras biológicas repetidas en distintos momentos del día, dado que este procedimiento sería muy invasivo y difícilmente justificable éticamente. Por ello, muchos estudios recurren al uso de tejidos accesibles o datos *postmortem*, en los que las muestras proceden de distintos individuos con características diferentes y el momento exacto de extracción suele ser desconocido o incierto.

Ante estas limitaciones, es necesario recurrir a métodos que permitan estimar el orden temporal relativo de las muestras como paso previo a la caracterización de los patrones rítmicos en la expresión génica. En este contexto, este trabajo compara dos de las metodologías más utilizadas para esta tarea, CIRCUST y CYCLOPS, aplicadas a un conjunto de datos de expresión génica en tejido muscular humano. A partir del orden temporal estimado, se procede a caracterizar el ritmo de expresión de cada gen mediante el ajuste de un modelo Cosinor, estimando parámetros como la amplitud, mesor y acrofase. Esta caracterización permite clasificar los genes en rítmicos o no rítmicos. Posteriormente, se evaluará la utilidad del orden estimado mediante la aplicación de diferentes algoritmos supervisados de *machine learning* (ML), con el objetivo de predecir la ritmicidad génica y comparar los resultados obtenidos en función del enfoque utilizado.

Palabras clave: ritmos circadianos, algoritmos, genes, *machine learning*, expresión génica , estimación del orden temporal.

Abstract

Circadian rhythms in living organisms are governed by an internal synchronization system that regulates gene expression in cycles of approximately 24 hours. These rhythms are essential for numerous physiological processes, making their study highly relevant in contemporary biomedical research. However, the analysis of circadian gene expression in humans poses significant challenges. A major limitation lies in the impossibility of collecting repeated biological samples from the same individuals at different times of day, as this would be highly invasive and ethically difficult to justify. As a result, many studies rely on accessible tissues or postmortem data, where samples are derived from different individuals with varying characteristics and the precise time of extraction is often unknown or uncertain.

Given these limitations, it is necessary to employ methods that can estimate the relative temporal ordering of samples as a preliminary step to characterizing rhythmic patterns in gene expression. In this context, the present work compares two of the most widely used methodologies for this task—CIRCUST and CYCLOPS—applied to a gene expression dataset from human muscle tissue. Based on the estimated temporal order, the rhythmic expression of each gene is characterized using a Cosinor model, from which parameters such as amplitude, mesor, and acrophase are estimated. This characterization enables the classification of genes as rhythmic or non-rhythmic. Subsequently, the utility of the estimated temporal order is evaluated through the application of various supervised machine learning (ML) algorithms aimed at predicting gene rhythmicity and comparing the results obtained under each methodological approach.

Keywords: circadian rhythms, algorithms, genes, machine learning, gene expression, temporal order estimation.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1: Introducción

1.1. Introducción a la expresión génica

La expresión génica es el proceso por el cual las células utilizan la información almacenada en los genes para fabricar moléculas que les permiten funcionar correctamente. Este proceso se desarrolla en dos etapas: en la transcripción, la información del ADN se copia en una molécula de RNA mensajero (RNA-m); en la traducción, el RNA-m se utiliza como plantilla para sintetizar proteínas. En la Figura 1.1 se esquematiza el proceso de expresión génica.

En contextos de análisis de datos, la expresión génica se mide cuantificando los niveles de RNA-m presentes en las células. Esta medición, obtenida mediante técnicas como la secuenciación, proporciona datos numéricos que permiten estudiar la actividad relativa de los genes en distintas condiciones.

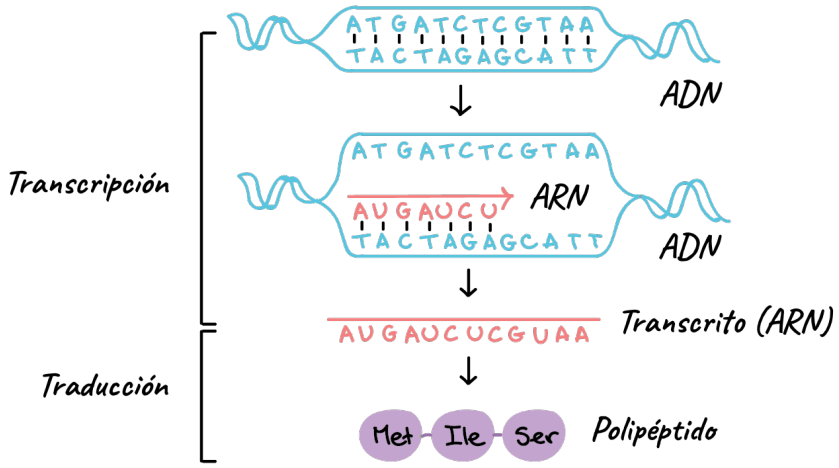

Figura 1.1: Etapas de la expresión génica: transcripción y traducción. La transcripción es el proceso en el que la célula copia la secuencia de DNA (del inglés *Deoxyribonucleic Acid*) en una molécula de RNA-m, reemplazando la timina (T) por uracilo (U). En la traducción, el ARNm es leído por los ribosomas, que lo interpretan en tripletes llamados codones. Cada codón codifica un aminoácido específico, formando una cadena polipeptídica que dará lugar a una proteína funcional. Imagen obtenida de [1].

Aunque el RNA-m no siempre refleja directamente la cantidad de proteína producida

debido a mecanismos reguladores adicionales, su cuantificación es ampliamente aceptada como aproximación del nivel de expresión génica. Para corregir posibles discrepancias y obtener mediciones más precisas, los datos de expresión deben ser normalizados. Este proceso resulta esencial no solo para reducir el impacto de factores técnicos o condiciones experimentales heterogéneas entre muestras, sino también para garantizar que los valores obtenidos representen con mayor fidelidad la verdadera expresión génica. Una revisión más detallada sobre los métodos de normalización puede consultarse en [2].

La expresión génica no ocurre de manera aleatoria, sino que está finamente regulada para garantizar que los genes se activen en el momento adecuado. Un caso particular es el de los genes circadianos, que se activan y desactivan según la hora del día.

1.2. Importancia de los genes circadianos

Los ritmos circadianos son ciclos biológicos internos que regulan nuestro organismo a lo largo del día. Estos ritmos se controlan mediante un reloj biológico interno, cuyo período aproximado es de un día, de ahí viene su nombre circadiano o 'cerca de un día'. El reloj circadiano permite sincronizar los ritmos de diferentes funciones fisiológicas (sueño, temperatura corporal o estado de ánimo) de los seres vivos mediante factores externos como la luz solar, la temperatura, la actividad física o los hábitos alimenticios que engloban el ciclo natural de aproximadamente 24 horas [3].

En mamíferos, el reloj circadiano se organiza de una manera jerárquica. En la cima de esta jerarquía, encontramos un reloj biológico central en el núcleo supraquiasmático (SCN, *suprachiasmatic nucleus*, por sus siglas en inglés) del hipotálamo anterior, que se encarga de coordinar los ritmos biológicos de todo el organismo. Por otra parte, en nuestro organismo existen diferentes tejidos y órganos como el hígado, el corazón o el cerebro que poseen su propio reloj biológico local. Esto explica por qué algunos genes pueden mostrar ritmicidad en ciertos tejidos, pero en otros no. Estos relojes periféricos, junto al SCN, permiten una sincronización coherente en todo el organismo, asegurando que las funciones fisiológicas se ajusten adecuadamente al entorno y condiciones ambientales [4]. A nivel molecular, la coordinación del reloj circadiano se refleja en patrones temporales rítmicos de expresión génica, específicos de cada tejido [5].

El reloj circadiano no solo sincroniza los ritmos de los órganos y tejidos del cuerpo, sino que también regula la actividad de ciertos genes cuya expresión sigue un ciclo de aproximadamente 24 horas. Estos genes, llamados genes circadianos, desempeñan un papel fundamental en la regulación y sincronización de funciones biológicas diarias en nuestro organismo. Dependiendo del momento del día, su actividad puede aumentar o disminuir, asegurando que cada proceso celular ocurra en el momento adecuado [6].

Esta ritmicidad está controlada por el reloj biológico interno o reloj circadiano. Se conoce que, en mamíferos, aproximadamente el 50 % de los genes presentan patrones de expresión rítmicos con una periodicidad cercana a 24 horas.

A nivel molecular, el funcionamiento del reloj circadiano depende de un grupo de aproximadamente 12 genes clave, conocidos como genes *core* [3, 5]. Estos genes forman el

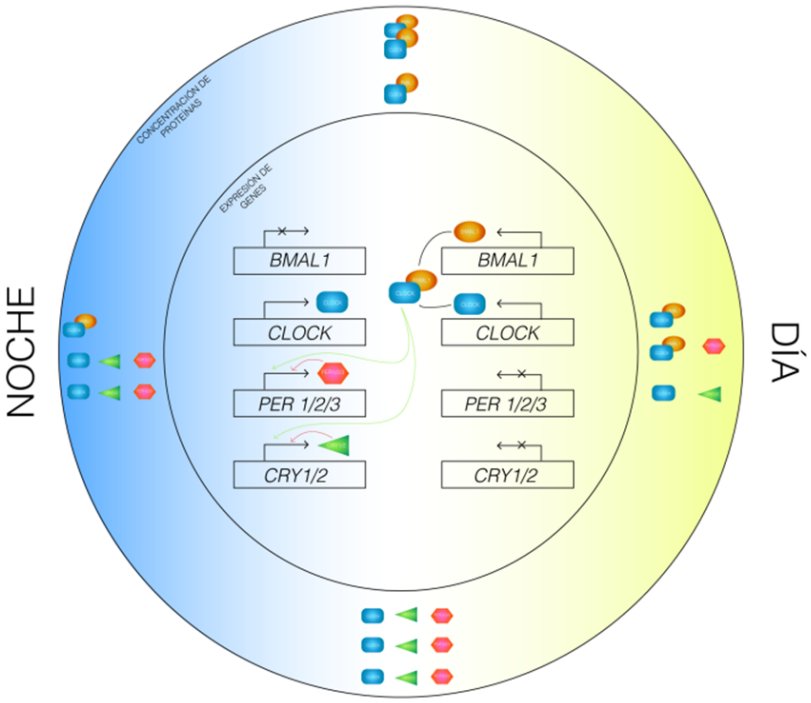

Figura 1.2: Representación esquemática del ciclo circadiano molecular de los genes *core*. Durante el día, el gen BMAL1 (óvalos naranjas) comienza su expresión y su proteína se acumula en el núcleo de la célula. Allí se une a la proteína CLOCK (rectángulos azules), que se produce de forma constante. Este complejo BMAL1–CLOCK, activan la transcripción de los genes PER 1/2/3 (hexágonos rosas) y CRY 1/2 (triángulos verdes), conocidos como genes represores. A medida que avanza el día, las proteínas PERs y CRYs se acumulan y forman complejos que entran al núcleo y bloquean el complejo activador BMAL1-CLOCK. Por la noche, los complejos PER–CRY son degradados por el sistema de eliminación celular, liberando la inhibición y permitiendo que BMAL1 reinicie su ciclo de expresión, cerrando así un bucle de regulación de aproximadamente 24 horas. Imagen obtenida de [7].

mecanismo central que genera y mantiene los ritmos circadianos endógenos en las células. Los principales genes *core* incluyen BMAL1, CLOCK, PERs y CRYs, los cuales interactúan en un ciclo de retroalimentación que regula la expresión de otros genes circadianos. Este ciclo funciona mediante dos fases principales, tal y como se ilustra en la Figura 1.2:

1. Fase activadora: Durante el día, los genes CLOCK y BMAL1 activan la expresión de otros genes, como PERs y CRYs.
2. Fase inhibidora: A medida que las proteínas PERs y CRYs se acumulan, inhiben la actividad de CLOCK y BMAL1, deteniendo su propia producción. Con el tiempo, estas proteínas se degradan, reiniciando el ciclo y completando un ritmo de aproximadamente 24 horas.

Este sistema de regulación circadiana permite que los distintos relojes periféricos de los

tejidos mantengan una sincronización con el reloj central del SCN, garantizando que las funciones fisiológicas ocurran de manera coordinada en todo el organismo.

El correcto funcionamiento de este sistema es esencial para la salud, ya que alteraciones en los genes *core* pueden afectar procesos metabólicos clave y aumentar el riesgo de enfermedades metabólicas (obesidad, diabetes tipo 2), enfermedades neurodegenerativas como el Alzheimer, o un mayor riesgo de padecer enfermedades inmunitarias como el cáncer [8, 9]. Además, su estudio ha permitido el desarrollo de estrategias en medicina personalizada, como la cronoterapia (administración de fármacos en los momentos más efectivos del día), la optimización de tratamientos quirúrgicos y la adaptación de dietas a los ritmos biológicos individuales [10].

1.3. Dificultades en el análisis de genes circadianos

El estudio de los genes circadianos en humanos enfrenta múltiples desafíos, entre los que destacan:

- **Dificultad para realizar muestreos secuenciales en humanos.** A diferencia de los estudios en animales, en los que es posible recoger muestras en distintos momentos del día mediante protocolos controlados, en humanos la obtención de biopsias seriadas en tejidos internos resulta altamente invasiva e inviable. Por este motivo, la mayoría de los estudios en humanos se basan en tejidos accesibles, como sangre o piel. Sin embargo, esto limita la aplicabilidad de los resultados a otros órganos con patrones de ritmicidad presumiblemente distintos [11].
- **Variabilidad biológica y factores ambientales.** Los ritmos circadianos pueden verse alterados por factores externos como la exposición a la luz artificial, el trabajo nocturno, la alimentación en horarios irregulares o el nivel de estrés. Estas variables pueden desincronizar los ritmos circadianos individuales y generar una gran variabilidad en los datos, dificultando la identificación de patrones consistentes entre diferentes sujetos. Además, las diferencias genéticas entre individuos pueden influir en la periodicidad y amplitud de los ritmos circadianos, reduciendo la capacidad de generalizar los hallazgos [12, 13].
- **Dificultad en la estimación del estado circadiano en muestras *postmortem*.** Las muestras *postmortem* se emplean frecuentemente en estudios circadianos debido a la dificultad de obtener biopsias seriadas en individuos vivos. Sin embargo, presentan un desafío fundamental: la incertidumbre sobre el momento exacto del fallecimiento y la fase del ciclo circadiano en la que se encontraba el individuo. Esta falta de una referencia temporal precisa complica la reconstrucción de los ritmos circadianos a partir de estos tejidos. Asimismo, tras la muerte, el RNA comienza a degradarse progresivamente, lo que afecta la calidad y fiabilidad de los datos transcriptómicos. La degradación *postmortem* puede alterar los niveles de expresión génica detectados, introduciendo ruido y sesgos que dificultan la identificación de verdaderos patrones rítmicos [14].
- **Patrones de expresión complejos y no sinusoidales.** La expresión de los genes circadianos no siempre sigue un patrón sinusoidal regular, sino que puede presentar

formas asimétricas y complejas. Esto dificulta la modelización de los datos y la identificación de los ritmos circadianos, especialmente cuando los datos son ruidosos o incompletos [15].

Para evitar estas dificultades, muchos estudios han recurrido a trabajar con modelos de animales como ratones y babuinos [16]. Estos modelos permiten realizar muestreos en múltiples momentos del día bajo condiciones controladas. Sin embargo, presentan limitaciones, ya que existen diferencias fisiológicas y conductuales con los humanos que pueden afectar la extrapolación de los resultados.

Mientras que en modelos animales es posible minimizar la variabilidad circadiana controlando factores como la luz, la alimentación o la actividad, en humanos este control es mucho más complicado, especialmente en tejidos internos. Por ello, muchos estudios en humanos recurren al análisis de muestras *postmortem* [17, 18], que permiten estudiar la expresión génica en distintos tejidos sin la necesidad de biopsias seriadas. No obstante, el uso de estas muestras introduce otros desafíos, como la dificultad para reconstruir la ritmicidad, al carecer de etiquetas temporales o ser imprecisas, añadiendo una capa de complejidad adicional en el análisis de expresiones circadianas.

Por ello, el estándar en estudios génicos humanos es disponer de datos de expresión *postmortem* para un mismo individuo en varios tejidos, para los que se desconoce el instante del día en que se tomaron las muestras, o bien se dispone de una estimación imprecisa. Debido a este motivo, surge la necesidad de desarrollar metodologías capaces de estimar el orden temporal de las muestras a partir de sus patrones de expresión génica.

1.3.1. Estimación del orden temporal

El problema de la estimación del orden temporal consiste en ordenar las muestras biológicas de forma que se recupere su dinámica circadiana implícita, permitiendo así identificar genes cuya expresión varía de manera rítmica a lo largo del día. Resolver este problema es fundamental para proseguir con análisis de ritmicidad a partir de patrones de expresión ordenados. Para abordar este problema, se han desarrollado diferentes estrategias, incluyendo enfoques basados en modelos estadísticos y métodos de aprendizaje automático.

Respecto de los patrones de expresión, uno de los métodos clásicos más empleados para el análisis de ritmos circadianos es el modelo Cosinor [19], que ajusta los datos a una función cosenoidal y permite estimar parámetros clave como la fase, la amplitud y el período del ritmo. Su simplicidad y robustez han hecho que sea ampliamente utilizado en estudios cronobiológicos. Sin embargo, su principal limitación es que asume que los ritmos circadianos son perfectamente sinusoidales y simétricos, lo que no siempre se ajusta a los datos biológicos reales, donde los patrones de expresión pueden ser asimétricos o presentar variaciones en la amplitud.

Para abordar esta limitación, se han desarrollado modelos más flexibles, como el *Frequency Modulated Möbius* (FMM) univariante [15], que permite modelar oscilaciones más complejas sin imponer restricciones estrictas sobre la periodicidad o la forma de la onda, permitiendo un análisis fisiológicamente más interpretable. A diferencia del modelo Co-

sinor, FMM puede capturar ritmos con asimetrías y variaciones en la amplitud, lo que lo hace más adecuado para analizar datos ruidosos o con estructuras circadianas irregulares. Su capacidad para adaptarse a formas de onda más realistas ha demostrado ser una ventaja en estudios donde la expresión génica circadiana no sigue un patrón estrictamente sinusoidal.

En este contexto, la identificación de ritmicidad génica es esencial para descifrar el papel funcional del reloj circadiano en la regulación génica. Detectar con precisión qué genes exhiben oscilaciones robustas a lo largo del día permite no solo construir mapas funcionales del ritmo biológico, sino también reconocer patrones alterados que puedan estar implicados en enfermedades o desajustes fisiológicos. Gracias a estos enfoques avanzados, es posible detectar ritmos circadianos en condiciones experimentales más variadas, permitiendo una caracterización más precisa de los genes con expresión rítmica, incluso en datos ruidosos o incompletos. La integración de metodologías clásicas con enfoques emergentes, como el aprendizaje automático, abre nuevas posibilidades para ampliar esta caracterización, proporcionando herramientas más flexibles y adaptativas ante la complejidad y variabilidad inherentes a los datos transcriptómicos humanos.

1.4. Objetivos del trabajo

En este trabajo se analizarán dos de las principales metodologías utilizadas en la reconstrucción del orden temporal, como son CIRCUST y CYCLOPS, y su aplicación en la identificación de genes circadianos.

Objetivo general

Describir, comparar y evaluar dos de los principales métodos aplicados a la reconstrucción del orden temporal de los ritmos moleculares en humanos: CIRCUST y CYCLOPS. Además, se aplicarán ambos métodos a datos reales de expresión génica con el objetivo de estudiar su rendimiento en la identificación de genes con comportamiento circadiano.

Objetivos específicos

- Describir los fundamentos teóricos de los métodos utilizados para la estimación del orden temporal en estudios circadianos.
- Comparar y evaluar el rendimiento de los dos principales métodos, analizando sus ventajas e inconvenientes en la detección de patrones rítmicos en datos de expresión génica.
- Aplicar y evaluar estos métodos estadísticos para la identificación de genes circadianos a partir de datos reales de expresión génica en tejido humano, concretamente en músculo esquelético.

- Analizar la precisión y robustez de los modelos en la reconstrucción de perfiles circadianos, considerando la presencia de ruido y datos incompletos.
- Evaluar el rendimiento de varios algoritmos supervisados de aprendizaje automático (regresión logística, SVM, *random forest* y *XGBoost*) en la clasificación de genes circadianos a partir de características derivadas de la expresión génica.

1.5. Asignaturas relacionadas para la elaboración del trabajo

- **Análisis de Datos, Análisis Multivariante y Técnicas de Aprendizaje Automático:** se han empleado modelos supervisados como la regresión logística, SVM, *random forest* y *XGBoost* para clasificar genes circadianos. Realizando una interpretación de los resultados obtenidos.
- **Inferencia Estadística y Modelos Lineales:** se han aplicado técnicas de validación, interpretación de coeficientes y evaluación de modelos.
- **Regresión y ANOVA:** se han utilizado técnicas para evaluar la significancia de los parámetros del modelo, aplicando criterios estadísticos como el p-valor y el ajuste de múltiples genes para identificar genes rítmicos.
- **Computación Estadística:** todo el análisis del trabajo se ha realizado con el software estadístico `R`.
- **Análisis de Series Temporales:** se ha aplicado el modelo Cosinor para detectar patrones cíclicos en la expresión génica ordenada temporalmente, con el fin de identificar genes con comportamiento circadiano.

1.6. Estructura de la memoria

Este documento se estructura de la siguiente forma:

- **Capítulo 1. Introducción:** presenta el contexto del trabajo, se detallan las principales dificultades del problema, se exponen los objetivos del proyecto y las asignaturas relacionadas.
- **Capítulo 2. Metodología:** describe en detalle la metodología empleada. Se explican los métodos utilizados para estimar el orden temporal, los modelos de ajuste de ritmicidad aplicados y los criterios de validación. También se incluye la metodología de clasificación mediante modelos supervisados para predecir si un gen es rítmico o no, así como las métricas utilizadas para evaluar su rendimiento. Por último, se explican los algoritmos supervisados utilizados en la clasificación de genes circadianos.
- **Capítulo 3. Resultados:** expone los resultados obtenidos a lo largo del trabajo, se presenta la evaluación de los modelos supervisados de clasificación utilizados y se comentan los resultados obtenidos.

- **Capítulo 4. Conclusiones:** recoge las conclusiones principales tras la realización del trabajo y se proponen posibles líneas futuras de investigación.
- **Bibliografía:** recoge los enlaces y referencias a los recursos utilizados.
- **Apéndice A. Metodología adicional:** contiene detalles técnicos complementarios que enriquecen la sección metodológica principal.
- **Apéndice B. Código fuente y *scripts*:** incluye los principales *scripts* programados en R, utilizados en el trabajo.

Capítulo 2: Métodos

Este capítulo introduce el problema estadístico de estimación del orden temporal en contextos donde no se dispone de información explícita sobre el instante de recogida de las muestras biológicas. A continuación, se detallan las dos principales metodologías existentes para abordar esta tarea, CIRCUST y CYCLOPS, exponiendo sus fundamentos teóricos, sus estrategias computacionales, y los motivos por los cuales resultan especialmente adecuados para el análisis circadiano en muestras humanas sin anotación horaria. Además, se propone una comparativa entre ellos, analizando sus diferencias metodológicas, ventajas y limitaciones. Por último, se comentan y explican los diferentes algoritmos supervisados utilizados para la clasificación de genes mediante las características derivadas de la expresión génica obtenido con CIRCUST.

2.1. El problema de estimación del orden temporal

Sea $X = [x_1, \ldots, x_m]$ una matriz de expresión génica de dimensión $G \times m$, donde:

- G es el número de genes.
- m es el número de muestras cuyos tiempos de recogida $(t_1, \ldots, t_m)$ son desconocidos.

Cada vector x_j representa los valores de expresión en los G genes para las muestras recogidas en el instante t_j. Por lo que x_{ij} representa el valor de expresión del gen i-ésimo para la muestra recogida en el instante de tiempo t_j.

El objetivo es encontrar un orden óptimo π de las muestras, de modo que refleje la progresión de la ritmicidad circadiana en los datos de expresión génica. Formalmente, buscamos una permutación:

$$\pi : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, m\}$$

tal que las muestras reorganizadas

$$X_\pi = [x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(m)}]$$

preserven la progresión temporal relativa de la expresión génica.

Para encontrar esta permutación π^*, se define una función de coste $C(X_\pi)$ que mide lo bien que la ordenación reconstruye la estructura circadiana. El objetivo es minimizar la siguiente función:

$$\pi^* = \arg\min_{\pi} C(X_\pi)$$

Esta función de coste puede definirse de distintas maneras, dependiendo del método utilizado.

Generalmente, este tipo de problemas pertenece a la categoría de problemas *NP-hard*. Para abordarlo, se han explorado distintas aproximaciones con un denominador común en todas ellas, que consiste en el uso de una circunferencia como estructura subyacente para la resolución del problema, donde los ritmos circadianos con una periodicidad de 24 horas se formulan de forma natural.

2.2. Métodos de estimación del orden temporal

Ante la falta de muestras con información temporal explícita, han proliferado métodos diseñados para abordar el problema de la estimación del orden temporal sin necesidad de anotaciones temporales de referencia.

Una estrategia común entre estos enfoques es la reducción de dimensionalidad, con el objetivo de recuperar una estructura circular latente, representativa del ritmo circadiano, que subyace en los datos de expresión génica. En muchos casos, dicha estructura se modeliza como una circunferencia que representa el ciclo completo del ritmo circadiano, y su estimación se lleva a cabo mediante técnicas que combinan estadística multivariante y redes neuronales autoasociativas (*autoencoders*), actuando como herramientas de reducción no lineal de la dimensión.

En los últimos años han emergido métodos específicamente desarrollados para reconstruir la ritmicidad circadiana mediante datos con orden desconocido, entre los que destacan:

- **CIRCUST:** se basa en estadística circular, propone una reducción de dimensionalidad mediante un Análisis de Componentes Principales Circular (CPCA, del inglés *Circular Principal Component Analysis*), y está optimizado para proporcionar resultados robustos en muestras humanas *postmortem* [3].
- **CYCLOPS:** combina técnicas de reducción de dimensionalidad con un *autoencoder* circular, forzando la representación latente de las muestras en una trayectoria elíptica coherente con el ritmo circadiano [20].

A continuación, se detalla cómo se resuelve el problema de estimación del orden temporal mediante los métodos CIRCUST y CYCLOPS.

2.2.1. CIRCUST

CIRCUST (CIRCular-robUST) [3] es una metodología basada en estadística circular y análisis de ritmos, desarrollada para reconstruir el orden temporal de muestras biológicas con tiempos de recogida desconocidos y modelizar patrones de ritmicidad circadiana en datos de expresión génica. CIRCUST permite inferir tanto la posición relativa de cada muestra dentro del ciclo circadiano como los parámetros que caracterizan la ritmicidad de los genes expresados en distintos tejidos. Además, ha sido diseñado para ser robusto frente al ruido y la variabilidad entre individuos, lo que lo hace especialmente útil para estudios transcriptómicos en humanos.

Metodología

La metodología CIRCUST detallada en [3] se compone de dos grandes bloques:

1. **Reconstrucción del orden temporal: Circular PCA**
   CIRCUST utiliza una técnica de reducción de dimensionalidad no lineal, denominada CPCA, para identificar la estructura circular subyacente en los datos de expresión génica.

   En primer lugar, se parte de una submatriz compuesta por genes *core*, cuya ritmicidad ha sido ampliamente validada en mamíferos. A partir de este conjunto se calculan dos componentes principales (*eigengenes*) que capturan los principales patrones de variación circadiana presentes en los datos de expresión. Estos *eigengenes* se proyectan sobre el círculo unitario, de forma que a cada muestra se le asigna un ángulo que representa su posición relativa dentro del ciclo circadiano. Esta representación angular permite inferir un orden circular entre las m muestras, entendiendo como tal una disposición secuencial en el espacio circular $[0, 2\pi)$.

   Este concepto de orden circular es esencial en el algoritmo CIRCUST, ya que, debido a la naturaleza periódica del espacio angular, un mismo conjunto de fases puede dar lugar a hasta $2m$ ordenaciones temporales distintas, al existir ambigüedad tanto en el punto de inicio como en la dirección (horaria o antihoraria). Por ello, uno de los pasos críticos del método es la sincronización del orden estimado, con el fin de fijar una referencia temporal coherente y orientada biológicamente. La Figura 2.1 ilustra este procedimiento: CIRCUST parte de perfiles de expresión génica desordenados en el espacio euclídeo y aplica una estimación del orden temporal en el espacio circular mediante un enfoque de reducción de dimensionalidad no lineal basado en CPCA. Tras la etapa de sincronización, la parte derecha de la figura muestra la reconstrucción temporal de las expresiones génicas.

   De forma complementaria, la técnica CPCA empleada en esta etapa no solo permite estimar la posición relativa de las muestras, sino que también constituye una herramienta eficaz para la detección de muestras atípicas (*outliers*), a partir del análisis geométrico de su proyección en el espacio de componentes principales.

2. **Identificación de genes rítmicos y ajuste de modelo**

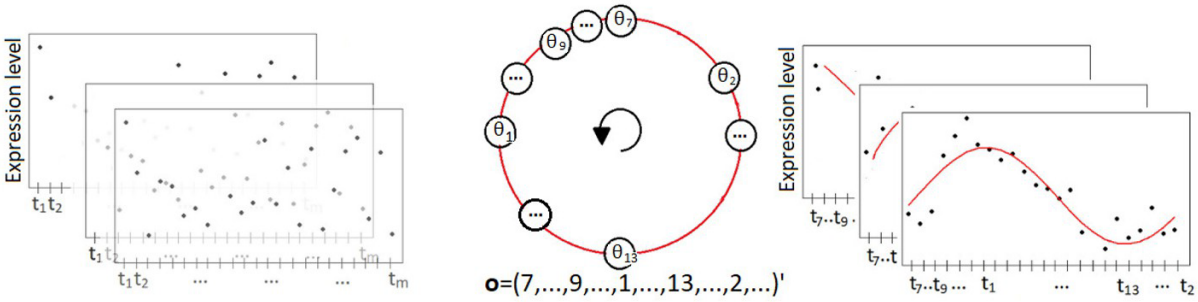

Figura 2.1: Esquema del problema de estimación del orden circular. Izquierda: datos de expresión desordenados a lo largo de 24 horas. Centro: orden circular obtenido con sentido antihorario. Derecha: expresión génica ordenada a lo largo de 24 horas. Imagen obtenida de [3].

Una vez estimado el orden temporal de las muestras, se procede a la normalización de los datos de expresión génica y al filtrado de aquellos genes con niveles de expresión muy bajos o patrones atípicos. Posteriormente, se identifican los genes que presentan una ritmicidad circadiana más pronunciada en el tejido analizado. Con el objetivo de incrementar la robustez del análisis, este procedimiento se repite sobre múltiples subconjuntos aleatorios de genes rítmicos, evaluando en cada iteración la calidad del ajuste y agregando los resultados obtenidos. Esta estrategia permite una caracterización más fiable y resistente al ruido del ritmo de expresión circadiana.

En el presente trabajo, el objetivo principal es obtener una estimación del orden temporal de las muestras utilizando un conjunto reducido de genes *core*, que funcione como referencia estándar para la comparación con otros algoritmos competidores en tejido muscular humano. A diferencia de la metodología original de CIRCUST, no se implementa el proceso completo de agregación robusta dirigido a caracterizar exhaustivamente la ritmicidad de un conjunto de genes característico de cada tejido. En su lugar, se adopta una estrategia simplificada y adaptada, centrada exclusivamente en la obtención del mejor ordenamiento posible a partir de ejecuciones parciales del procedimiento. Esta adaptación permite preservar la esencia del enfoque original y reducir su complejidad computacional.

Validación

Antes de aplicar un algoritmo en datos sin anotación temporal, es fundamental evaluar su desempeño en entornos controlados donde se dispone de información horaria conocida. Una de las métricas estándar más utilizadas para evaluar el rendimiento de los métodos de estimación del orden temporal, cuando se dispone de etiquetas temporales reales, es el error absoluto medio (MAE, por sus siglas en inglés, *Mean Absolute Error*) entre el orden real y el orden estimado. En la práctica, se consideran aceptables desviaciones de ± 3 horas, y para su cálculo generalmente se requiere rescalar el orden estimado al intervalo [0,24) [3].

CIRCUST ha sido validado en distintos contextos experimentales que incluyen datos humanos (epidermis, músculo esquelético y corteza prefrontal) y de modelo animal (tejidos

de babuino), todos con información temporal conocida, lo que ha permitido evaluar la precisión del orden reconstruido. Ha demostrado una notable capacidad para recuperar el orden relativa de las muestras y reproducir patrones circadianos coherentes, incluso en condiciones de alta variabilidad biológica o en tejidos con ritmicidad compleja.

2.2.2. CIRCUST adaptado

A continuación, se describen las etapas necesarias para abordar el problema de estimación del orden temporal, siguiendo una adaptación simplificada de la metodología propuesta por CIRCUST.

Antes de detallar las etapas, es conveniente introducir brevemente el contexto y la notación. Sea $[X]$ la matriz de expresión génica en crudo (*raw counts*). Durante la etapa de preprocesamiento, se filtran los genes no expresados y se identifican posibles muestras atípicas u *outliers*. El resultado de este paso es una nueva matriz $[N]$, que contiene los datos de expresión ya depurados y listos para el análisis. A partir de ella, los datos se normalizan y se aborda el problema de estimación del orden temporal, obteniendo como resultado la matriz $[X_\pi]$ de muestras ordenadas cuyas etapas principales se resumen a continuación:

Preprocesado de datos

La entrada a esta etapa es la matriz $[X]$ de datos crudos y desordenados de expresiones de un tejido. Se realizan los siguientes pasos:

1. Se eliminan los genes con más de un 30 % de las muestras con valores de expresión nulos, al considerarse poco informativos para el análisis.
2. Se aplica un filtrado robusto por nivel de expresión utilizando una media recortada por la derecha, eliminando únicamente el 10 % de los valores más altos de cada gen (Sección A.1 del Apéndice). A continuación, se calcula la media sobre el 90 % restante. Este enfoque permite mitigar el efecto de valores extremos elevados en genes poco informativos. Aquellos genes cuya media recortada se sitúa por debajo de un umbral predefinido, que depende del estudio, son excluidos del análisis. La elección del umbral se realizará de forma empírica en cada conjunto de datos garantizando una proporción elevada y estable en distintos conjuntos conocidos de genes rítmicos.
3. La matriz resultante se normaliza al intervalo $[-1, 1]$ mediante escalado *min–max*.

De esta primera etapa se obtiene la matriz $[N]$ de datos de expresión génica normalizados.

Ordenación y sincronización

La matriz de *input* de esta etapa es $[N]$, la matriz normalizada y procesada. Se realizan cinco pasos principales:

1. Se realiza la selección del conjunto de genes *core* extraídos de [3], que incluye: *PER1, PER2, PER3, CRY1, CRY2, ARNTL, CLOCK, NR1D1, RORA, DBP, TEF* y *STAT3*. La elección de estos genes se fundamenta en su buena ritmicidad circadiana en una amplia variedad de tejidos humanos, como se evidencia en [5]. Debido a su gran comportamiento rítmico, han sido utilizados como genes de referencia en numerosos estudios centrados en el análisis de la expresión circadiana [20–22].

   A partir de este subconjunto, se aplica el algoritmo CPCA sobre la submatriz de $[N]$, compuesta únicamente por los 12 genes seleccionados. La salida de CPCA proporciona un primer orden estimado, que posteriormente se utiliza para reordenar la matriz completa de expresión génica.

2. Se lleva a cabo un procedimiento de detección y eliminación de *outliers* mediante dos enfoques:

   a) **Identificación basada en CPCA:** a partir del análisis CPCA, se calcula la distancia euclídea al origen de cada muestra en el espacio bidimensional definido por los dos primeros eigengenes. Se clasifican como *outliers* aquellas muestras cuya distancia euclídea supera el umbral de 2 unidades respecto al origen del círculo. Este umbral que podría variar entre estudios identifica muestras con comportamiento atípico fuera de la estructura circular esperada.

   b) **Identificación mediante residuos estandarizados:** se calculan los residuos estandarizados (véase en la Sección A.2 del Apéndice) de cada muestra a partir del ajuste de un modelo Cosinor, detallada en la Sección 2.3. Se consideran *outliers* aquellas muestras cuyos residuos estandarizados se encuentran fuera del intervalo $[-3, 3]$.

3. Una vez eliminados los *outliers* comunes entre ambos enfoques, se aplica el algoritmo CPCA sobre la submatriz de genes core de $[N]$, obteniéndose un nuevo vector de orden circadiano. Este nuevo orden estimado se utiliza para reordenar la matriz completa de expresión génica.

4. Se lleva a cabo un proceso de sincronización entre los genes. Es importante recordar que el orden obtenido admite $2m$ configuraciones temporales distintas. Para fijar un punto de inicio, se utiliza como referencia el gen *ARNTL*, cuya ritmicidad está bien caracterizada en la literatura. Se ajusta un modelo Cosinor sobre su perfil de expresión, y se estima la fase correspondiente a su pico de máxima expresión. A continuación, se aplica un desfase circular al orden estimado de las muestras para que dicho pico se alinee en π. Esta elección se justifica porque, según [23], el gen *ARNTL* alcanza su máxima expresión antes del inicio del periodo inactivo, tradicionalmente representado por π.

   Para determinar la orientación del orden (horaria o antihoraria), se consideran los momentos de máxima expresión de los genes *PERs* y *CRYs*, estimados mediante el ajuste de modelos Cosinor individuales. Tal como se recoge en la literatura, estos genes alcanzan su máximo durante el periodo de luz, es decir, antes del pico de expresión de *ARNTL* (π). Por tanto, se selecciona como orientación válida aquella que sitúe la mayor proporción de estos genes dentro del intervalo $(0, \pi]$, en coherencia con su fase biológica esperada.

5. Se recupera el nuevo vector de fases ajustadas y se utiliza para reordenar la matriz de expresión génica.

De esta segunda etapa se obtiene la matriz $[X_\pi]$ de datos de expresión génica ordenados y sincronizados.

2.2.3. CYCLOPS

El algoritmo CYCLOPS integra tres componentes fundamentales: la detección de patrones estructurales en los datos de expresión, la incorporación de información sobre la conservación evolutiva de los genes, y una arquitectura basada en *autoencoders* circulares. Esta combinación le permite reconstruir una trayectoria cerrada que representa la dinámica circadiana de las muestras, sin necesidad de conocer sus tiempos reales de recogida.

CYCLOPS sigue una secuencia de pasos para ordenar las muestras a lo largo de un ciclo de 24 horas, estimando así su fase circadiana relativa:

Preprocesado de datos

Se lleva a cabo una limpieza inicial del conjunto de datos en la que se eliminan, por un lado, las muestras del individuo con datos incompletos (mencionado previamente) y, por otro, aquellas identificadas como *outliers* según el análisis realizado con el método CIRCUST, con el fin de garantizar una comparación coherente y libre de sesgos por valores atípicos.

Este paso es la única modificación del algoritmo original que se ha hecho en este trabajo.

Ordenación

1. Se aplica una descomposición en valores singulares (SVD, *Singular Value Decomposition*, por sus siglas en inglés) para reducir la dimensión. Esta técnica extrae los principales componentes de variabilidad, denominados *eigengenes*, que resumen patrones globales de expresión (ver Sección A.4 del Apéndice).

2. A partir de los *eigengenes*, se entrena un *autoencoder* cuyo objetivo es reconstruir los datos originales a partir de una representación intermedia de menor dimensión. La característica principal de esta arquitectura es que la capa intermedia (*bottleneck*) se ha diseñado específicamente para capturar la naturaleza circular del problema, representando cada muestra como un ángulo sobre una elipse. La red tiene tres capas:

   - **Capa de Entrada:** los *eigengenes* reducidos.
   - **Capa oculta circular (*bottleneck*):** codifica cada muestra como un ángulo.

- **Salida:** reconstruye los *eigengenes* originales.

El entrenamiento se realiza mediante descenso de gradiente con momento, lo que mejora la estabilidad y la velocidad de convergencia. Para más detalles sobre esta arquitectura, puede consultar la Sección A.6 del Apéndice.

3. Tras el entrenamiento, cada muestra se proyecta sobre un ángulo entre 0 y 2π, que representa su fase circadiana estimada. Esta coordenada refleja su posición relativa dentro del ciclo biológico de 24 horas.

Validación

Aunque CYCLOPS ha demostrado buen rendimiento en múltiples estudios, su validación en humanos es más limitada que la de CIRCUST. La mayoría de sus aplicaciones se han centrado en tejidos de ratón, donde se ha comprobado una alta concordancia entre las fases estimadas y los tiempos reales. En humanos, los resultados más destacados provienen de estudios *postmortem* en corteza prefrontal y en piel, donde se ha logrado estimar la fase circadiana con errores medios aceptables. Para mejorar la robustez del algoritmo en estos casos, se restringió el análisis a genes homólogos a aquellos con ritmicidad circadiana bien caracterizada en múltiples tejidos de ratón, aprovechando la conservación evolutiva del sistema circadiano.

En la Tabla 2.1 se han resumido las principales características, ventajas y limitaciones de los métodos de estimación del orden presentados en este trabajo.

2.2.4. Otros métodos

Estos métodos requieren que las muestras estén etiquetadas con la anotación temporal de recogida. Utilizan esta información para modelizar directamente la relación entre la expresión génica y el momento del ciclo circadiano.

Ejemplos que destacan en esta metodología son: Molecular Timetable [24], un método pionero que emplea patrones horarios predeterminados de expresión génica para realizar predicciones del estado circadiano mediante regresión. Por otro lado, BIO_CLOCK [25] utiliza redes neuronales profundas entrenadas con datos de expresión génica anotados para obtener predicciones del tiempo circadiano. Más recientemente, TimeSignature [26] utiliza métodos supervisados basados en *machine learning*, específicamente algoritmos de *Random Forest*, para seleccionar genes con alta capacidad predictiva en la estimación del tiempo circadiano. Estos genes seleccionados actúan como marcadores biológicos, proporcionando una predicción estable y precisa incluso con un número reducido de muestras.

Estas metodologías proporcionan herramientas precisas y computacionalmente eficientes. Sin embargo, requieren conjuntos de datos con tiempos circadianos conocidos previamente, limitando su aplicación práctica, particularmente en estudios genéticos en humanos donde las etiquetas temporales son inciertas o desconocidas.

Aspecto	**CYCLOPS**	**CIRCUST**
Particularidades	▪ Conocimiento previo de genes rítmicos en ratón (conservación evolutiva). ▪ Detecta trayectorias elípticas.	▪ Estadística circular para estimar fase. ▪ Apto para muestras *postmortem* (ARN degradado). ▪ Genes *core* definidos por el usuario; sin depender de otras especies. ▪ Tolera muestreo no uniforme del ciclo.
Ventajas	▪ Robusto y eficiente en grandes cohortes humanas. ▪ Validado en varios tejidos y poblaciones.	▪ Muy robusto al ruido; mejora la interpretación de datos ruidosos. ▪ Facilita la evaluación de *outliers* y la selección de *eigengenes*. ▪ Lista *core* adaptable a distintos contextos biológicos.
Desventajas	▪ Precisión dependiente de información externa para fijar la hora absoluta. ▪ Ordenamientos relativos; requiere datos completos de 24 h. ▪ Difícil cuantificar la influencia de *outliers*. ▪ El uso de la conservación evolutiva puede favorecer la captura de ritmos influenciados por el entorno, alineados con el ciclo luz/oscuridad, en lugar de ritmos circadianos endógenos.	▪ Frecuencia de muestreo baja puede afectar la exactitud temporal. ▪ Sensible a la elección del conjunto inicial de genes *core*.

Tabla 2.1: Comparativa entre los algoritmos CYCLOPS y CIRCUST.

2.3. Modelización e identificación de ritmos circadianos

La identificación de genes con expresión rítmica es una etapa fundamental en el análisis transcriptómico circadiano, ya que permite estudiar el reloj biológico, detectar biomarcadores y comparar condiciones fisiológicas. Para ello, se han desarrollado distintos enfoques estadísticos, tanto paramétricos como no paramétricos, cada uno con sus propias ventajas y limitaciones.

Entre los métodos no paramétricos, uno de los más utilizados es JTK_CYCLE [27], que se basa en estadísticos de orden y pruebas de correlación (tipo *Kendall*) para detectar patrones oscilatorios sin asumir una forma funcional concreta de la señal. Este tipo de enfoques destaca por su robustez ante ruido y su aplicabilidad en situaciones con escasas réplicas, aunque a menudo presentan menor potencia para detectar ritmos con formas no canónicas o baja amplitud.

Por otro lado, los métodos paramétricos, que parten de un modelo explícito para la forma de la señal circadiana. El más representativo es el modelo Cosinor [19]. Por su simplicidad y utilidad, en la práctica es el modelo más extendido en la literatura y sirve como estándar en numerosas aplicaciones. El modelo Cosinor es un modelo de regresión no lineal que se formula de la siguiente manera:

$$x(t_j) = M + A \cdot \cos(\omega t_j + \phi) + \varepsilon_j$$

donde:

- $x(t_j)$: valor de expresión génica observado en el instante t_j,
- M: *Mesor*, valor medio de la oscilación,
- A: *Amplitud*, magnitud del ritmo circadiano,
- $\omega = \frac{2\pi}{T}$: frecuencia angular del ciclo, siendo $T = 24$ horas para ritmos circadianos,
- ϕ: *Acrofase*, momento del pico máximo de expresión (en radianes),
- ε_j: término de error asociado a la muestra j.

Se ilustran las características rítmicas de Cosinor en la Figura 2.2.

Este modelo permite no solo describir la dinámica rítmica de la expresión génica, sino también evaluar estadísticamente si dicha ritmicidad es significativa. En particular, el problema de detectar si un gen presenta un patrón circadiano puede formularse como un contraste de hipótesis sobre la amplitud (A):

$$H_0 : A = 0 \quad \text{(no hay ritmo)} \qquad \text{frente a} \qquad H_1 : A > 0$$

Aunque el modelo Cosinor se expresa inicialmente de forma no lineal, puede reparametrizarse para expresarse en términos de un modelo lineal a partir de identidades trigonométricas básicas. Esto permite aplicar contrastes tipo *F de Fisher* para comparar el modelo completo con otro que no incluye componente oscilatoria en términos del parámetro A. La formulación y detalles sobre la resolución de este contraste se detalla en la

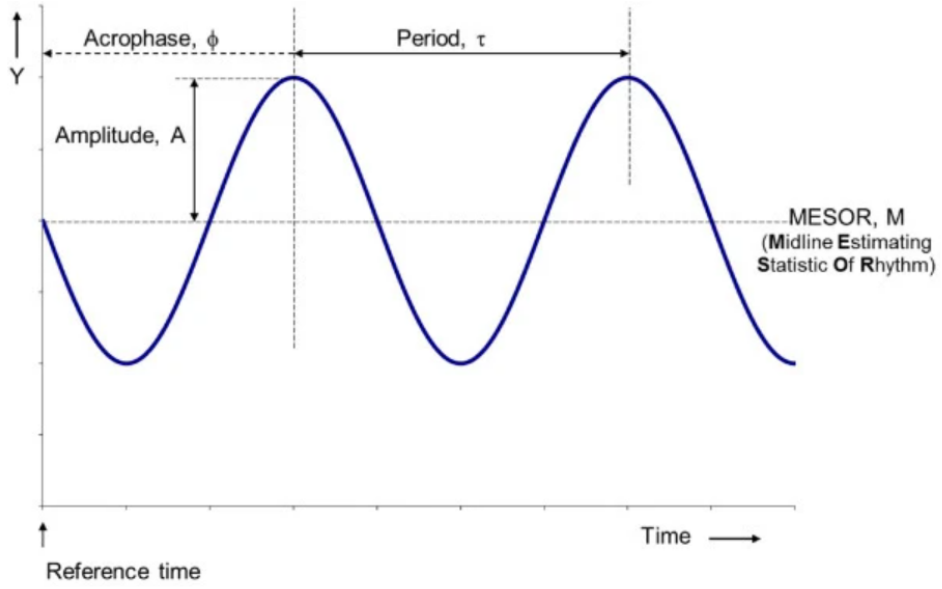

Figura 2.2: Definición de las características rítmicas de Cosinor. Imagen obtenida de [19].

Sección A.3 del Apéndice.

Dado que se realizan contrastes de hipótesis para un gran número de genes de forma simultánea, es necesario aplicar una corrección por comparaciones múltiples para evitar un exceso de resultados falsamente significativos. En este trabajo se utiliza el procedimiento de Benjamini-Hochberg [28], que controla la tasa de falsos descubrimientos (FDR), es decir, la proporción esperada de falsos positivos entre los genes declarados significativos. Tras aplicar esta corrección, se considera que un gen muestra ritmicidad estadísticamente significativa si su p-valor ajustado es inferior a 0.05.

Este marco del modelo Cosinor ha servido de base para numerosas herramientas desarrolladas para la detección de ritmicidad circadiana. En particular, los paquetes `cosinor` y `cosinor2` de *R* implementan específicamente contrastes F para este modelo, permitiendo evaluar la significación de la componente oscilatoria. Además, paquetes más completos como MetaCycle, RAIN, BIO CYCLE o DiffCircaPipeline integran el modelo Cosinor (y algunas de sus extensiones), junto con otros métodos no paramétricos, para estimar parámetros clave (como amplitud, fase y período) y comparar perfiles rítmicos entre condiciones experimentales.

Evaluación de la ritmicidad

Como medida de bondad de ajuste para evaluar la calidad del ajuste del modelo a los datos se propone el coeficiente de determinación R^2. Este estadístico para el gen i, se define como:

$$R_i^2 \;=\; 1 \;-\; \frac{\sum_j \left(X_{ij} - \widehat{X}_{ij}\right)^2}{\sum_j \left(X_{ij} - \overline{X}_i\right)^2},$$

donde:

- Y_{ij} es la expresión observada del gen i en la muestra j.
- $\widehat{Y}_{ij}$ es el valor predicho por el ajuste Cosinor para el gen i en la muestra j.
- $\overline{Y}_i = \frac{1}{m}\sum_j Y_{ij}$ es la media de las observaciones del gen i a lo largo de las m muestras.

Esta medida indica qué proporción de la variabilidad observada en los datos de expresión es explicada por el modelo. Valores cercanos a 1, reflejan una buena concordancia entre el modelo y los datos experimentales, mientras que valores bajos indican un ajuste pobre. En el contexto del análisis de expresiones circadianas, sujetas a gran variabilidad, valores de R^2 entre 0.5 y 0.8 son indicadores de una adecuación moderada-alta del ajuste a los datos.

En este trabajo, se ha utilizado el coeficiente de determinación R^2 en dos contextos distintos. Por un lado, se ha calculado el R^2 de los genes *core* circadianos como medida complementaria para evaluar la calidad del orden estimado por los algoritmos. Además de utilizarse sobre el conjunto de 54 genes principales. Por otro lado, el R^2 también se ha empleado como medida de bondad de ajuste en el modelo Cosinor aplicado individualmente a cada gen, permitiendo evaluar el grado de ritmicidad del perfil de expresión.

2.4. Clasificación supervisada de genes rítmicos

En este trabajo, además de estimar el orden temporal de las muestras, se aborda la identificación de genes con expresión circadiana mediante un enfoque de clasificación supervisada. Para ello, se asigna a cada gen una etiqueta binaria que indica la presencia o ausencia de ritmicidad, y se entrenan varios algoritmos de aprendizaje automático para predecir dicha etiqueta a partir de características cuantitativas derivadas del ajuste del modelo Cosinor a los datos de expresión génica. Este enfoque permite evaluar la capacidad predictiva de distintos modelos y explorar su utilidad en la detección automatizada de genes rítmicos.

Formalmente, se plantea un problema de clasificación supervisada en el que cada gen se representa mediante un par $(\boldsymbol{x}_i, y_i)$, donde $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ corresponde al conjunto de características de expresión génica utilizadas, y $y_i \in \{0, 1\}$ indica si el gen presenta ritmicidad circadiana. El objetivo es predecir la clase y_i exclusivamente a partir del patrón de expresión, utilizando algoritmos de aprendizaje automático.

Se han entrenado y evaluado cuatro modelos de aprendizaje supervisado: Regresión Logística, Árboles de Decisión (*Random Forest*), Máquinas de Vectores de Soporte (SVM, *Support Vector Machine*, por sus siglas en inglés) con *kernel* radial, y *Extreme Gradient Boosting* (*XGBoost*). Esta selección se justifica por la diversidad de enfoques y nive-

les de complejidad que ofrecen, lo que permite comparar el rendimiento desde modelos lineales simples hasta técnicas avanzadas basadas en árboles. Además, se trata de algoritmos ampliamente consolidados en la literatura y habitualmente empleados en tareas de clasificación binaria.

Para su implementación se ha utilizado el lenguaje y entorno estadístico **R**, mediante los paquetes `caret`, `e1071`, `randomForest` y `xgboost`. En el Sección A.8 se describen las funciones utilizadas, así como los principales hiperparámetros y argumentos necesarios para reproducir cada modelo.

En modelos con múltiples hiperparámetros configurables, es habitual dividir los datos en tres subconjuntos: entrenamiento, validación y *test*. El conjunto de entrenamiento se utiliza para ajustar el modelo, el de validación para seleccionar la mejor configuración de hiperparámetros, y el de *test* para evaluar el rendimiento final. No obstante, cuando el tamaño del conjunto de datos es limitado, resulta más eficiente emplear estrategias como la validación cruzada, que reutilizan los datos disponibles y permiten obtener estimaciones de rendimiento más estables y representativas.

En este trabajo se ha aplicado validación cruzada interna de cinco particiones (*5-fold cross-validation*). Este procedimiento divide el conjunto de entrenamiento en cinco bloques de igual tamaño. En cada una de las k iteraciones, uno se reserva para validación y los cuatro restantes se utilizan para entrenar. De este modo, cada observación actúa como muestra de validación exactamente una vez, y el modelo se entrena cinco veces. Las métricas de validación se obtienen como la media de las cinco iteraciones, proporcionando una estimación más robusta del rendimiento del modelo.

Una vez completada esta validación interna, se evalúa sobre el conjunto de *test*. Esta evaluación final permite estimar la capacidad de generalización del modelo a nuevas muestras y cuantificar su utilidad real. Este procedimiento se ha aplicado en todos los modelos considerados. Además de las métricas de clasificación globales, se ha utilizado la matriz de confusión para obtener una visión más detallada del comportamiento de cada modelo.

A continuación se presentan los modelos evaluados, describiendo en cada caso los fundamentos metodológicos que explican su funcionamiento, así como la selección y el impacto de los hiperparámetros ajustados. Cabe señalar que en esta sección puede producirse un abuso de notación, en el sentido de que algunos símbolos utilizados previamente se reutilizan con otro significado. Esta práctica se adopta por conveniencia expositiva, y el contexto debería permitir al lector interpretar correctamente cada caso.

2.4.1. Regresión Logística

La regresión logística es un modelo lineal ampliamente utilizado en problemas de clasificación binaria [29]. Se enmarca dentro de los modelos lineales generalizados (*Generalized Linear Models*, GLM), concretamente en la familia binomial, y emplea como función de enlace el *logit*. Su finalidad es modelizar la probabilidad de que una observación pertenezca a una de dos clases posibles, a partir de una combinación lineal de variables explicativas. Esta combinación se transforma mediante la función logística (*sigmoide*),

que restringe el rango de salida al intervalo $[0, 1]$, permitiendo así su interpretación como una probabilidad. En tareas de clasificación binaria, la versión utilizada sin regularización permite interpretar directamente los coeficientes estimados como logaritmos del cociente de probabilidades, manteniendo una mayor transparencia en la relación entre predictores y variable respuesta.

Sea $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_m)^\top$ el vector de parámetros del modelo (*intercept* β_0 y m coeficientes), y sea $\pi_i = \Pr(y_i = 1 \mid \mathbf{x}_i)$ la probabilidad de que el gen i sea rítmico, dado su vector de características $\mathbf{x}_i$. El modelo logístico tiene la siguiente forma:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i. \tag{1}$$

Aplicando la función inversa del *logit*, obtenemos la expresión explícita de la probabilidad π_i:

$$\pi_i = \frac{1}{1 + \exp\left[-\left(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i\right)\right]}. \tag{2}$$

La decisión final sobre la clase predicha se realiza aplicando un umbral de decisión τ, habitualmente fijado en 0.5.

$$\hat{Y}_i = \begin{cases} 0, & \text{si } \hat{\pi}_i < \tau \quad \text{(gen no rítmico)}, \\ 1, & \text{si } \hat{\pi}_i \geq \tau \quad \text{(gen rítmico)}. \end{cases} \tag{3}$$

2.4.2. *Random Forest*

El algoritmo *Random Forest* (RF) es un método de aprendizaje automático basado en técnicas de ensamblado (*ensemble methods*), concretamente en el enfoque de *bootstrap aggregating* o *bagging* [30]. Consiste en construir un conjunto de árboles de decisión entrenados sobre muestras *bootstrap* del conjunto de datos original. Cada árbol se ajusta de forma independiente a su muestra remuestreada, lo que contribuye a reducir la varianza del estimador global.

En tareas de clasificación, la predicción final se obtiene mediante votación mayoritaria entre todos los árboles. Para incrementar la diversidad entre los árboles y disminuir la correlación entre ellos, en cada nodo se selecciona aleatoriamente un subconjunto de variables candidatas sobre el que se busca la mejor división. Esta combinación de remuestreo y selección aleatoria de variables mejora la capacidad de generalización del modelo y lo hace robusto frente al sobreajuste. La Figura 2.3 ilustra esquemáticamente el funcionamiento del algoritmo.

En el caso de tareas de clasificación binaria, cada árbol del RF actúa como un clasificador que construye una estructura jerárquica a partir de un nodo raíz, generando divisiones

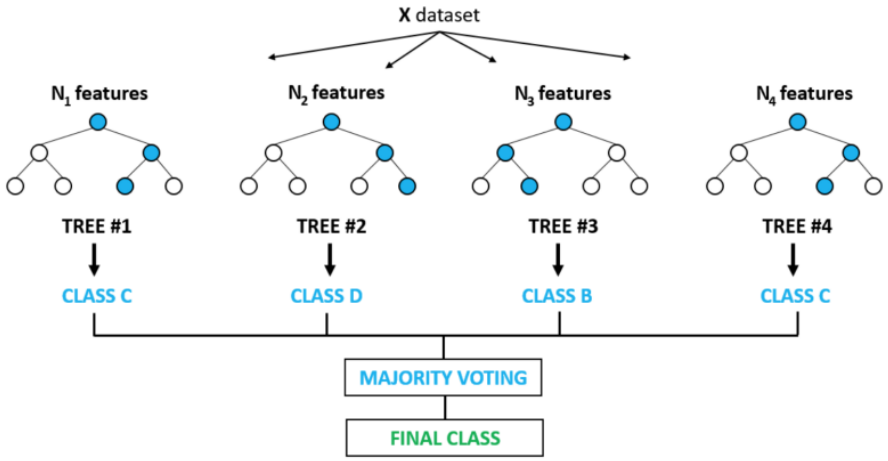

Figura 2.3: Ejemplo de un *ensemble* compuesto por cuatro árboles de decisión, es decir, un RF. Imagen obtenida de [30].

binarias sucesivas en función de las variables predictoras. En cada nodo interno, se selecciona la variable y el punto de corte que maximizan un criterio de pureza de la partición, como el índice de Gini o la entropía. El proceso continúa hasta alcanzar nodos terminales (hojas), donde se asigna la clase más frecuente entre las observaciones que llegan a dicha hoja.

Aunque los árboles individuales son modelos simples e interpretables, presentan una elevada varianza, especialmente si se permiten estructuras profundas. El uso combinado de múltiples árboles entrenados sobre muestras *bootstrap* mitiga esta desventaja mediante agregación.

Cada árbol se entrena sobre aproximadamente dos tercios de los datos originales (observaciones *in-bag*), mientras que el tercio restante (*out-of-bag*, OOB) actúa como conjunto de validación interna. El error OOB proporciona una estimación casi insesgada del error de generalización sin necesidad de aplicar validación cruzada.

2.4.3. Máquinas de Vectores Soporte

Las Máquinas de Vectores Soporte (*Support Vector Machines*, SVM) son algoritmos de aprendizaje supervisado, orientados tanto a tareas de clasificación como a regresión [31]. Su principio consiste en hallar el hiperplano que mejor separa las clases en el espacio de características. Cuando la tarea es de clasificación, se trata de maximizar la distancia (margen) entre las muestras de distintas clases. Solo las observaciones más próximas a esa frontera, llamados vectores soporte (*support vectors*), intervienen en la construcción del modelo, lo que le confiere gran capacidad de generalización y lo hace resistente tanto al sobreajuste como a la presencia de *outliers*.

En la formulación lineal estricta se busca el hiperplano $\theta^\top x + \theta_0 = 0$ con $\|\theta\| = 1$ que maximice la distancia M entre las clases:

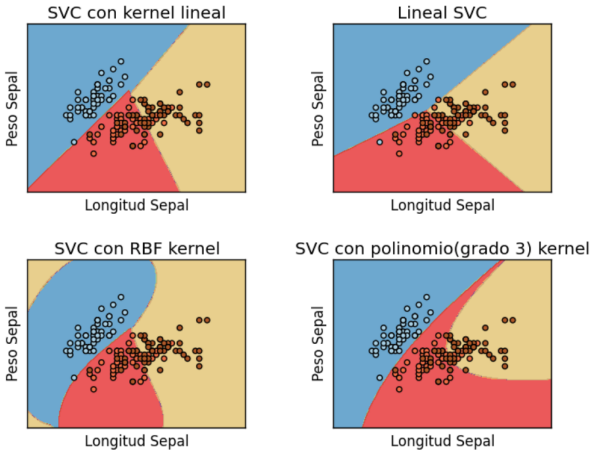

Figura 2.4: Representación gráfica del resultado de un modelo clasificador SVM sobre las tres clases de flores Iris setosa (azul), virgínica (amarillo) y versicolor (rojo) a partir del peso y la longitud del sépalo. Arriba izquierda: *kernel* lineal. Arriba derecha: sin *kernel*. Abajo izquierda: *kernel* RBF. Abajo derecha: *kernel* polinómico de grado 3. Imagen obtenida de [32].

$$\begin{aligned} \underset{\theta,\theta_0,\|\theta\|=1.}{\text{máx}} \quad & M \\ \text{sujeto a} \quad & y_i\big(\theta^\top x_i + \theta_0\big) \;\geq\; M, \quad i = 1, \ldots, m, \end{aligned}$$

Cuando las clases no son linealmente separables en el espacio original, se proyectan a un espacio de mayor dimensión mediante una función $\phi(\cdot)$. El producto interno $\phi(x_i)^\top \phi(x_j)$ se sustituye por un *kernel* $K(x_i, x_j)$, evitando calcular ϕ explícitamente. Los núcleos más habituales son:

$$\begin{aligned} \text{Lineal:} \quad & K(x_i, x_j) = x_i^\top x_j, \\ \text{Polinómico:} \quad & K(x_i, x_j) = \big(\gamma\, x_i^\top x_j\big)^d, \\ \text{RBF:} \quad & K(x_i, x_j) = \exp\big[-\gamma\, \|x_i - x_j\|^2\big], \end{aligned} \tag{8}$$

donde $\gamma > 0$ controla la amplitud del núcleo y d es el grado polinómico.
Además, se puede introducir un parámetro de penalización $C > 0$ que controla el compromiso entre la anchura del margen y el error de entrenamiento: cuando C es pequeño se impone una regularización fuerte, permitiendo que más puntos queden dentro del margen. Por el contrario, valores grandes de C suavizan la regularización y el modelo prioriza clasificar correctamente las muestras de entrenamiento, a costa de estrechar el margen. Este parámetro permite un mayor control del sobreajuste del modelo. El parámetro γ, por su parte, determina la influencia de cada punto de entrenamiento sobre la frontera de decisión. Un γ pequeño genera superficies suaves y de baja complejidad, mientras que valores altos pueden derivar en modelos con alta varianza y bajo poder de generalización.

Se puede apreciar gráficamente un ejemplo de clasificación utilizando los diferentes tipos de *kernel* explicados en la Figura 2.4.

En este trabajo se ha empleado un *kernel* radial (*Radial Basis Function*, RBF), especialmente útil cuando se desconoce la estructura real de separación entre clases, ya que permite modelizar relaciones altamente no lineales sin requerir un ajuste excesivo del modelo.

2.4.4. *Extreme Gradient Boosting*

Extreme Gradient Boosting (*XGBoost*) es un algoritmo de aprendizaje supervisado que pertenece a la familia de métodos de *Gradient Boosting* [33]. Su estrategia consiste en construir de forma secuencial y aditiva un conjunto de árboles de decisión, donde cada nuevo árbol corrige los errores residuales de los anteriores. A diferencia del *bagging* (como *Random Forest*), que entrena árboles de manera independiente para promediar sus predicciones, el *boosting* ajusta cada árbol teniendo en cuenta la información aprendida hasta el momento.

Formalmente, en cada iteración t, el algoritmo añade un nuevo árbol f_t al modelo mediante la minimización de una función objetivo que combina una función de pérdida diferenciable con un término de regularización:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t), \tag{1}$$

donde ℓ representa la función de pérdida (por ejemplo, log-loss en clasificación binaria), y $\Omega(f_t)$ penaliza la complejidad del nuevo árbol f_t. La optimización de esta expresión se realiza utilizando gradientes de primer y segundo orden, lo que permite una convergencia más eficiente y otorga mayor peso a las observaciones peor predichas. En *XGBoost*, $\Omega(f_t)$ suele adoptar la forma:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2, \tag{2}$$

donde T es el número de hojas del árbol, w_j el valor asignado a cada hoja, y γ, λ son hiperparámetros que controlan la penalización L_1 y L_2, respectivamente.

De forma predeterminada, *XGBoost* incluye regularización L_1 y L_2 como parte de su diseño, controladas mediante los hiperparámetros `alpha` y `lambda`, respectivamente. Ambos pueden ajustarse o anularse según el grado de penalización deseado. También incorpora técnicas adicionales para evitar el sobreajuste, como el *shrinkage* (reducción del peso de cada árbol mediante un factor de aprendizaje η) y el submuestreo tanto de observaciones como de características (*column subsampling*). Estas estrategias refuerzan la capacidad de generalización del modelo, especialmente en contextos con ruido o alta dimensionalidad, como ocurre con los datos de expresión génica.

Entre los hiperparámetros más relevantes se encuentra el número total de iteraciones que determina la cantidad total de árboles generados en el *ensemble*: un número elevado

puede mejorar la precisión, pero también incrementar el riesgo de sobreajuste. La tasa de aprendizaje controla cuánto influye cada nuevo árbol; valores bajos generan ajustes más progresivos y estables, mientras que tasas más altas aceleran la convergencia pero aumentan la sensibilidad al ruido. En cuanto a la profundidad máxima de los árboles base define su capacidad para capturar relaciones no lineales: árboles más profundos aumentan la complejidad del modelo, pero también su varianza. Además, *XGBoost* incorpora submuestreo tanto de observaciones como de variables, técnicas que introducen diversidad entre árboles, reducen la correlación entre ellos y favorecen la generalización, especialmente en entornos de alta dimensionalidad como el análisis de expresión génica.

2.5. Métricas de validación para los algoritmos de ML

A continuación se describen las métricas utilizadas para evaluar el rendimiento de cada modelo: precisión global (*accuracy*), sensibilidad, especificidad, precisión global balanceada y AUC (área bajo la curva ROC, del inglés *Area Under the ROC Curve*). Estas métricas, permiten evaluar la capacidad de cada modelo para distinguir correctamente entre genes rítmicos y no rítmicos, incluso en contextos de clases desbalanceadas.

Con el fin de facilitar la interpretación de las métricas de evaluación utilizadas en esta sección, se presenta en la Figura 2.5 la estructura estándar de una matriz de confusión para problemas de clasificación binaria. Dicha matriz resume las cuatro posibles salidas del modelo:

- VP: verdaderos positivos (rítmicos correctamente clasificados)
- VN: verdaderos negativos (no rítmicos correctamente clasificados)
- FP: falsos positivos (no rítmicos clasificados como rítmicos)
- FN: falsos negativos (rítmicos clasificados como no rítmicos)

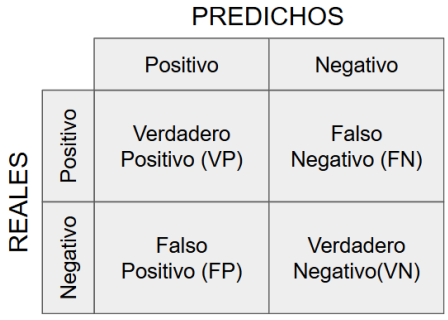

Figura 2.5: Matriz de confusión que muestra la distribución de predicciones en una tarea de clasificación binaria. Elaboración propia.

Esta representación constituye la base para el cálculo de las siguientes métricas:

- **Exactitud (*Accuracy*)**: mide la proporción de predicciones correctas sobre el total de muestras.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

- **Sensibilidad**: mide la proporción de casos positivos correctamente predichos.

$$\text{Sensibilidad} = \frac{VP}{VP + FN}$$

- **Especificidad**: mide la proporción de casos negativos correctamente predichos.

$$\text{Especificidad} = \frac{VN}{VN + FP}$$

- **Exactitud balanceada (*Balanced Accuracy*)**: corresponde al promedio entre la sensibilidad y la especificidad. Es especialmente útil cuando las clases están desbalanceadas.

$$\textit{Accuracy balanceado} = \frac{\text{Sensibilidad} + \text{Especificidad}}{2}$$

- **AUC**: representa el área bajo la curva ROC (*Receiver Operating Characteristic*), la cual muestra gráficamente la relación entre la sensibilidad (tasa de verdaderos positivos) y la tasa de falsos positivos (1 - especificidad) para diferentes umbrales de decisión. Esta curva se construye al variar el umbral de clasificación y calcular estos dos indicadores en cada punto. El valor del AUC, comprendido entre 0 y 1, resume la capacidad global del modelo para discriminar correctamente entre clases. Cuanto mayor sea el AUC, mejor será el rendimiento del modelo, siendo 1 el valor óptimo y 0.5 el rendimiento esperado por un clasificador aleatorio. En la Figura 2.6 se representa gráficamente un ejemplo genérico de una curva ROC.

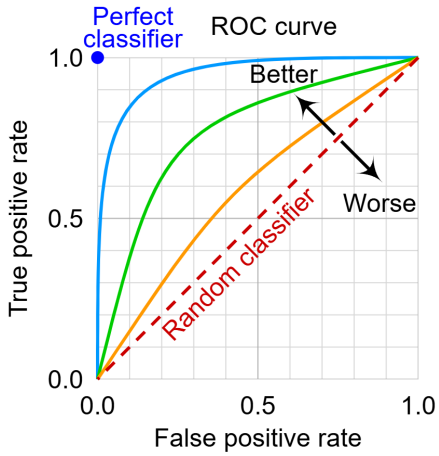

Figura 2.6: Curva ROC mostrando la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos. La línea roja diagonal indica el rendimiento de un clasificador aleatorio. Se ilustran tres clasificadores más en orden de rendimiento de menor a mayor (naranja, verde y azul). El punto azul indica el clasificador perfecto con un valor de 1. Imagen obtenida de [34].

Además, se han generado las correspondientes matrices de confusión con el objetivo de analizar detalladamente el rendimiento de los modelos. Dado que se trata de un problema de clasificación binaria, estas matrices presentan una estructura de dos filas y dos columnas, donde las filas representan las clases reales y las columnas las clases predichas.

Capítulo 3: Resultados

En este capítulo se utilizan datos con etiquetas temporales de expresión génica obtenidos a partir de biopsias de músculo esquelético humano [35]. El objetivo principal es reproducir la metodología descrita en la sección anterior para este conjunto de datos. En primer lugar se compara el rendimiento de los métodos CIRCUST y CYCLOPS a la hora de reconstruir el orden temporal de las muestras cuando la información horaria no está disponible. Una vez ordenadas, se abordará el problema de identificación de ritmicidad mediante el ajuste del modelo Cosinor. Por último, se evalúa la capacidad de distintos modelos de aprendizaje automático supervisado para discriminar genes rítmicos y no rítmicos a partir de sus perfiles de expresión temporal ordenados.

3.1. Descripción de los datos

Se analizan datos de expresión de músculo esquelético humano obtenidos mediante biopsias del músculo vasto lateral (*vasctus lateralis*) en 10 voluntarios sanos, registrando muestras de RNA cada 4 horas a lo largo de un periodo de 24 horas. Posteriormente, estas muestras fueron analizadas mediante secuenciación de RNA-seq. El número total de genes cuantificados fue de 14,415. Las características de los participantes se recogen en la Tabla 3.1. Las muestras se obtuvieron en condiciones de laboratorio estrictamente controladas, siguiendo un protocolo diseñado específicamente para minimizar la influencia de factores de confusión, como el estrés, la ingesta reciente de alimentos, la actividad física o la irregularidad en los ciclos de sueño. Este conjunto de datos se describe en [36] y ha sido utilizado en estudios similares [37, 38].

3.2. Estimación del orden temporal.

A continuación, se describen los resultados para la estimación del orden en las dos metodologías siguiendo los pasos descritos en la Sección 2.2.

3.2.1. Resultados de CIRCUST

El algoritmo *CIRCUST* está disponible en su versión original y completa en el repositorio público de GitHub [39]. La versión empleada en este trabajo corresponde a la adaptación

Donante	**Sexo**	**Edad (años)**	**IMC (kg/m²)**
I	M	22	28.3
II	F	37	26.4
III	M	24	25.3
IV	M	25	27.0
V	M	54	25.6
VI	M	53	26.8
VII	M	25	23.3
VIII	M	23	23.4
IX	M	21	20.9
X	M	30	22.4
Media ± sd	M=9, F=1	30 ± 10	24.1 ± 2.7

Tabla 3.1: Características principales de los 10 donantes utilizados para RNA-seq in vivo. Incluye información sobre su sexo (F=femenino; M=masculino), edad (años) y el índice de masa corporal (IMC) en kg/m^2. En la última línea se incluye la media y desviación típica (sd) de las variables cuantitativas. Tabla obtenida de [35].

explicada previamente, implementada en lenguaje `R`, cuya descripción detallada se incluye en la Sección B.2 del Apéndice.

Preprocesado de datos

En primer lugar, se realiza un análisis preliminar de los datos, donde se comprueba el número de muestras por paciente. Se decide descartar las muestras correspondientes al individuo VII debido a que están incompletas.

A continuación, se evalúan aquellos genes que presentan una proporción elevada de valores nulos en sus muestras, concretamente superior al 30 %. En este caso concreto, ningún gen supera ese umbral, por lo que no es necesario hacer este filtro.

Por otra parte, para eliminar genes no expresados o poco informativos, se aplica un filtrado basado en la media recortada por la derecha al 10 %, como se describe en la metodología. En concreto, se calcula la media de la expresión en cada gen tras eliminar el 10 % de los valores más altos, y se descartan aquellos cuya media recortada no alcanza un umbral mínimo. La elección de dicho umbral, cambia según el estudio. Para su elección, se analiza cómo varía el número de genes conservados al modificar dicho umbral, tanto en el conjunto de 54 genes rítmicos conocidos como en el subconjunto de 12 genes *core* representativos del reloj circadiano.

Tal y como se muestra en la Figura 3.1, en este caso se ha establecido un umbral de corte en 0.4 con el objetivo de lograr una proporción alta y equilibrada de genes conservados. Este valor se ha determinado tras analizar cómo afecta dicha decisión tanto a la conservación de los 12 genes *core* principales como a la proporción restante de los 54 genes rítmicos identificados en [23]. Aunque es esperable que no todos estos genes se

expresen rítmicamente en todos los tejidos, una fracción significativa debería mantenerse en el tejido utilizado.

Finalmente, la matriz de expresión resultante contiene 12,625 genes (87.6 %), lo que implica que se han descartado 1,790 genes mediante esta técnica.

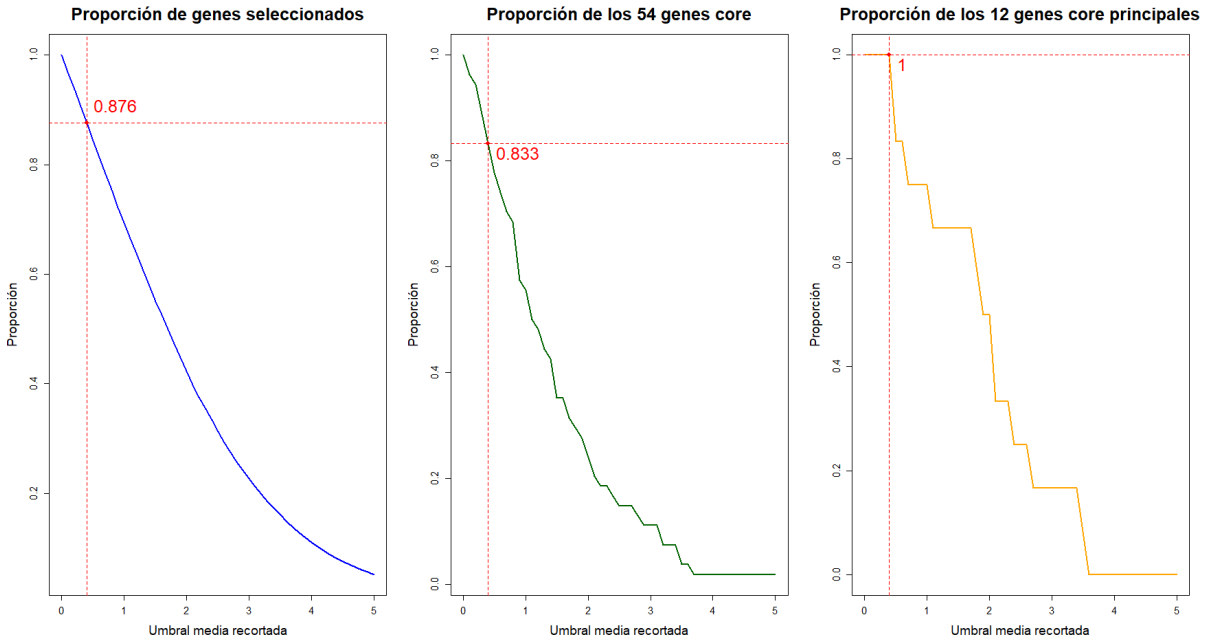

Figura 3.1: Gráficos que ilustran cómo varía la proporción de genes seleccionados para el análisis en función del umbral aplicado a la media recortada. Se indica en rojo el umbral seleccionado (0.4) y la proporción de genes que se conservan con ese umbral en distintos conjuntos de genes. Izquierda: conjunto de genes total. Medio: 54 genes típicamente rítmicos. Derecha: 12 genes *core*.

Ordenación y sincronización

Se aplica el método CPCA sobre la matriz de expresión de los genes *core*, previamente normalizada, con el objetivo de estimar el orden circadiano relativo de las muestras. Para evaluar si los dos primeros *eigengenes* capturan aproximadamente una estructura circular subyacente, se representan gráficamente en la Figura 3.2.

Aunque la representación directa de los dos primeros componentes principales (*eigengenes*) no permite identificar de forma clara una trayectoria circular en las muestras, se realiza una evaluación complementaria para verificar si describen un recorrido cíclico a lo largo del tiempo. La Figura 3.3 muestra dicha comprobación, donde se representa la trayectoria paramétrica tras ajustar ambos eigengenes mediante un modelo Cosinor. Esta figura confirma que ambos componentes realizan una trayectoria circular en el espacio bidimensional, validando así su capacidad para capturar la estructura rítmica subyacente en los datos.

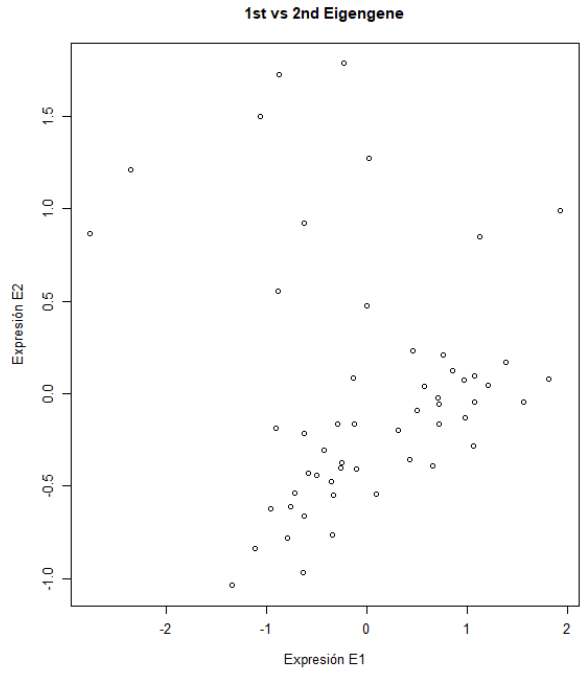

Figura 3.2: Representación de los 2 primeros *eigengenes* obtenidos de CPCA.

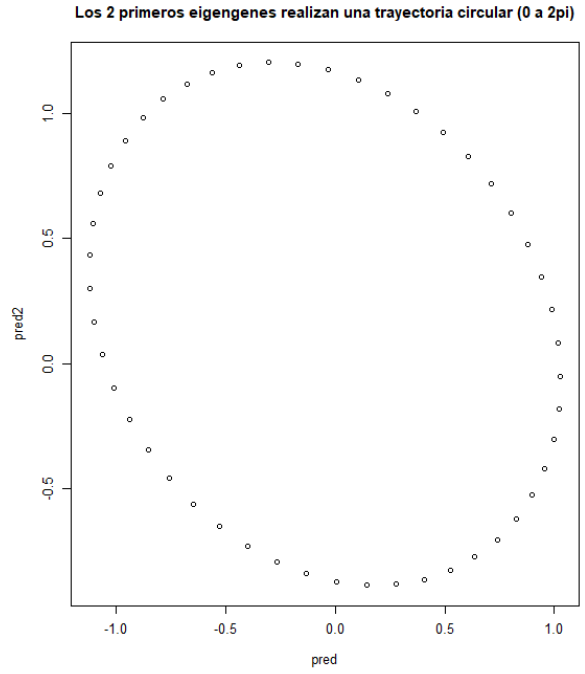

Figura 3.3: Representación de la trayectoria paramétrica (E1(t), E2(t)) a partir del ajuste Cosinor sobre los 2 primeros *eigengenes*.

En cuanto al análisis de valores atípicos, utilizando el criterio relacionado con los resultados del CPCA, se identifican como potenciales *outliers* las muestras *S12, S52 y S57* (Figura 3.4). De ellas, finalmente se eliminan *S52 y S57*, que muestran un residuo muy alto tras el ajuste cosinor. En el Sección A.7 del Apéndice se muestran estos ajustes.

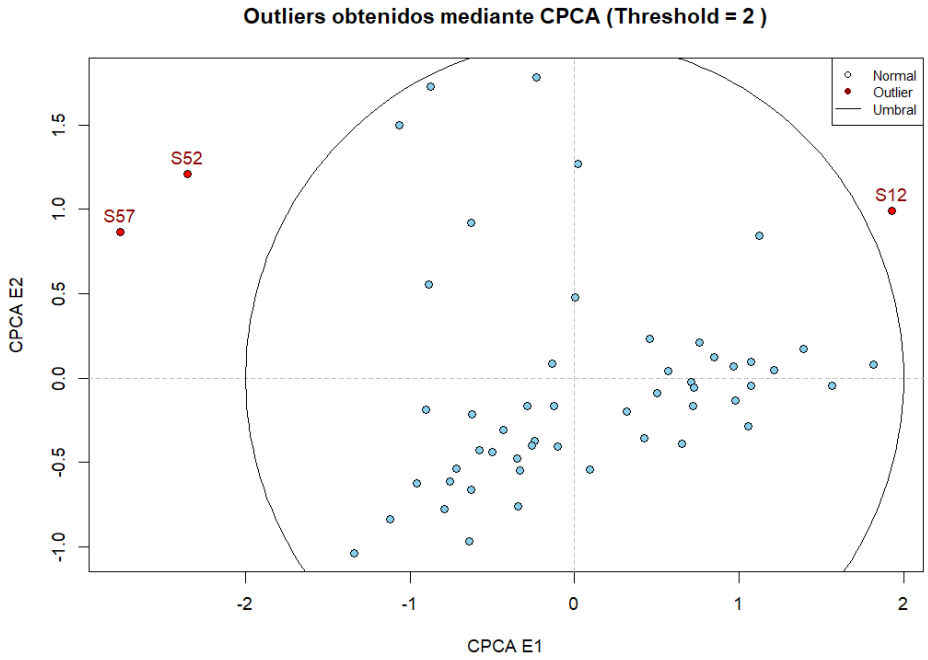

Figura 3.4: Visualización de outliers mediante el método CPCA. El círculo indica el umbral de detección y los puntos rojos son los *outliers* detectados.

Una vez detectadas y descartadas las muestras identificadas como *outliers*, los datos se normalizan nuevamente y se reordenan utilizando los 12 genes *core*, aplicando el procedimiento CPCA. El orden estimado se sincroniza tomando como referencia el gen *ARNTL*, uno de los reguladores centrales del reloj biológico. Siguiendo lo descrito en la metodología, el orden estimado se ajusta temporalmente para garantizar que el momento de máxima expresión del gen *ARNTL* se alinee con el valor π (véase en la Figura 3.5). A continuación, se determina la orientación (horaria o antihoraria) que maximiza la proporción de instantes de máxima expresión de los genes *PERs* y *CRYs* dentro del periodo diurno. El resultado de esta sincronización para los 12 genes core se muestra en la Figura 3.6. Esta transformación se aplica al resto de genes para los análisis posteriores.

Métricas de evaluación del orden temporal

Dado que en este conjunto de datos se disponen de las etiquetas temporales, como medida de precisión en la estimación del orden se propone el uso del MAE. En el caso de CIRCUST, se obtiene un error medio absoluto de 1.73 horas. La Figura 3.7 representa la concordancia entre el orden real de las muestras y el orden estimado con diferencias que en general no se exceden de las ± 3 horas recomendadas en la literatura.

De forma complementaria, tal y como se expone en la metodología, el coeficiente de determinación R^2 obtenido tras el ajuste del modelo Cosinor puede utilizarse como métrica para evaluar la coherencia del orden estimado. Dado que los genes *core* circadianos presentan, en general, un patrón rítmico bien definido en la mayoría de tejidos, es esperable que este comportamiento se vea reflejado en los valores del R^2. En la Tabla 3.2 se muestran

los valores del R^2 obtenidos para cada gen.

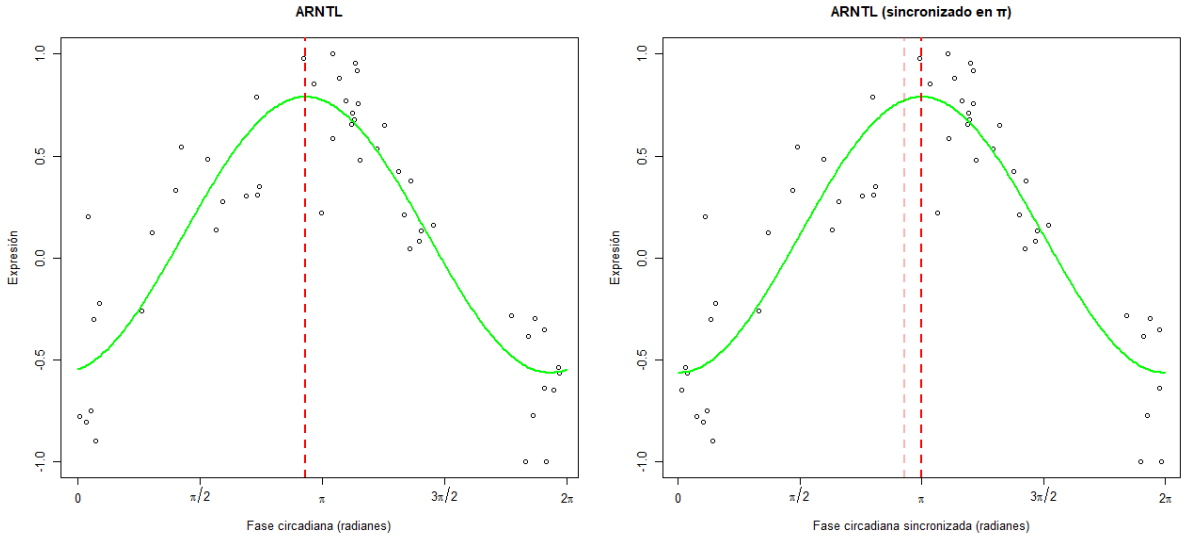

Figura 3.5: Visualización de la sincronización del gen *ARNTL*. En rojo se indica el pico de expresión máxima del gen. En verde se muestra los valores predichos del ajuste Cosinor.

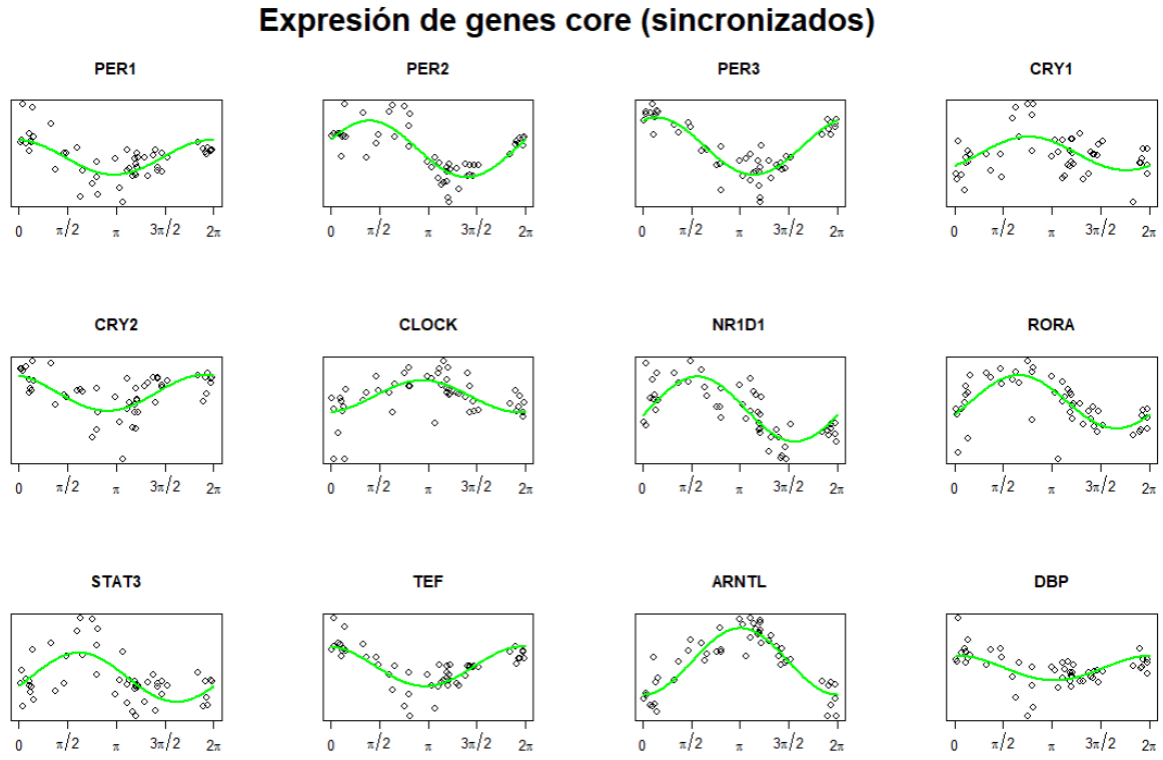

Figura 3.6: Patrones de expresión de los genes *core* sincronizados estimados por CIRCUST. En verde se muestra los valores predichos del ajuste Cosinor.

Como puede observarse, gran parte de los genes muestran un ajuste moderado-alto según

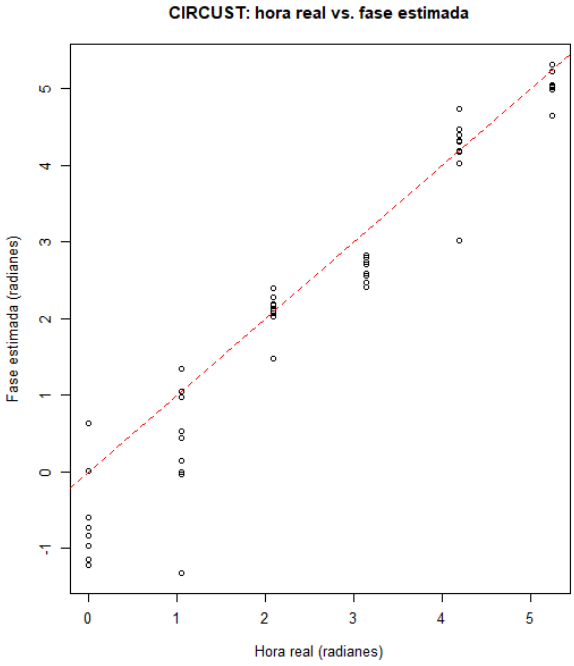

Figura 3.7: Relación entre el orden temporal real y estimado por CIRCUST. Se ha convertido el orden real de horas a radianes para facilitar la comparación con el orden estimado.

los umbrales de este estadístico en estudios de expresiones circadianas. En concreto, los genes *PER3* y *ARNTL* destacan por su fuerte ritmicidad, lo que concluye en un comportamiento muy marcado en este tejido. Por el contrario, genes como *CRY1* o *CLOCK* muestran una menor capacidad de ajuste, coherente con lo observado en la Figura 3.6.

Gen	R^2	**Gen**	R^2
PER1	0.483	NR1D1	0.575
PER2	0.677	RORA	0.474
PER3	0.823	STAT3	0.398
CRY1	0.265	TEF	0.640
CRY2	0.430	ARNTL	0.818
CLOCK	0.368	DBP	0.373

Tabla 3.2: Coeficiente de determinación R^2 del ajuste Cosinor para los 12 genes *core*.

3.2.2. Resultados de CYCLOPS

El algoritmo CYCLOPS, disponible en su repositorio original de *GitHub* [40], fue implementado íntegramente en el lenguaje *Julia v0.6*, una versión ya obsoleta que tiene más de seis años. Esta circunstancia impide su ejecución directa en las versiones actuales del lenguaje, lo que ha requerido una actualización del código. En concreto, ha sido necesario adaptar tanto funciones como estructuras de sintaxis desfasadas para garantizar su compatibilidad con *Julia v1.10*, la versión utilizada en este trabajo.

Preprocesado de datos

Como primer paso, se eliminan las muestras correspondientes al individuo VII, dado que presentan datos incompletos. Asimismo, se descartan los valores *outliers* detectados previamente mediante la metodología CIRCUST (*S52* y *S57*). Esta decisión permite asegurar una comparación coherente entre los algoritmos, utilizando exactamente las mismas muestras en todos los casos.

Para la ejecución del algoritmo, se opta por mantener los parámetros predeterminados establecidos en la implementación original, al considerarse adecuados y suficientemente contrastados para este tipo de datos.

Ordenación

El resultado del proceso se genera en tres etapas diferenciadas: la primera obtiene el orden circadiano estimado en radianes para cada muestra; la segunda recoge el orden de los índices correspondientes a dichas muestras; y la tercera proporciona la matriz de expresión génica completamente ordenada según el patrón circadiano inferido.

Tras analizar los resultados, se observa una distribución no uniforme en el círculo de la estimación en el orden temporal obtenido a partir de los 12 genes *core*. Tal y como se muestra en la Figura 3.8, la mayoría de las muestras aparecen concentradas en torno al ángulo $\frac{3\pi}{2}$, lo que indica una estimación deficiente del orden circadiano en este conjunto de datos. Este resultado no es fisiológicamente interpretable, ya que se pierde la variabilidad esperada en los patrones de expresión rítmica.

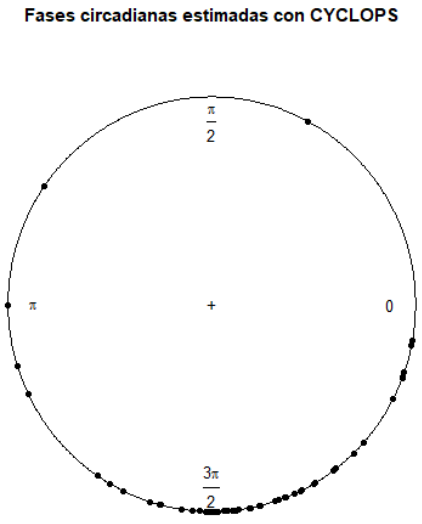

Figura 3.8: Distribución angular del orden temporal estimado por CYCLOPS.

Se ha abordado este problema explorando distintas configuraciones tanto en los parámetros del modelo como en la selección del conjunto de genes *core* utilizado como entrada. En concreto, se ha evaluado el conjunto de 17 genes *core* empleado en el repositorio ofi-

cial de *CYCLOPS 2.0* disponible en *GitHub* [41], así como el subconjunto de 10 genes propuesto por Talamanca en estudios previos [37, 42]. Adicionalmente, se han evaluado diversas configuraciones de los parámetros de ajuste de la metodología, con el objetivo de optimizar la calidad del orden temporal estimado y obtener distribuciones de fases más fisiológicamente coherentes.

Aunque las fases asignadas presentan ligeras variaciones en función del conjunto de genes *core* utilizado o de los parámetros empleados, en los escenarios considerados, las muestras continúan concentrándose en un único sector del círculo circadiano. Esta acumulación sugiere que persiste una limitación estructural en la capacidad del algoritmo para estimar adecuadamente el orden temporal en este conjunto de datos.

La Figura 3.9 ilustra los patrones de expresión de los 12 genes *core* tras la primera ordenación temporal obtenida mediante CYCLOPS, donde se evidencia la escasa dispersión temporal entre muestras.

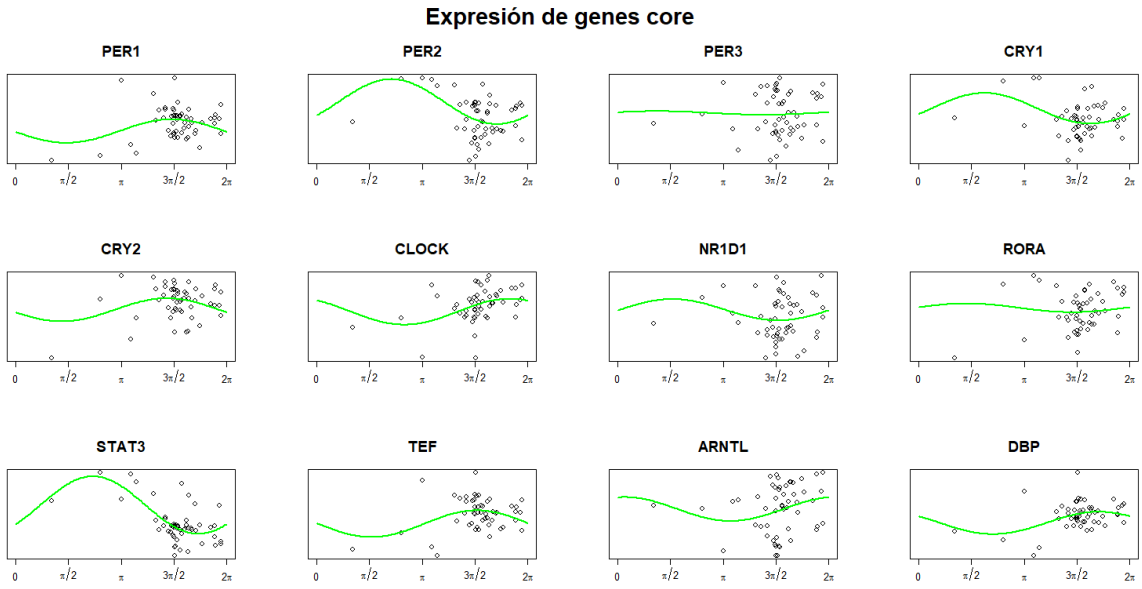

Figura 3.9: Patrones de expresión de los genes *core* estimadas por CYCLOPS. En verde se muestra los valores predichos del ajuste Cosinor.

Métricas de evaluación del orden temporal

Se han calculado distintas métricas para cuantificar la precisión del orden temporal estimado por CYCLOPS. En particular, el MAE entre el orden real y el estimado alcanza un valor de 5.65 horas, significativamente superior al umbral generalmente aceptado de ± 3 horas y al obtenido con la metodología *CIRCUST* (1.73 horas). La Figura 3.10 ilustra esta discrepancia, reflejando una menor concordancia entre los órdenes y un ajuste circadiano menos preciso.

Adicionalmente, se calcula el coeficiente de determinación R^2 resultante del ajuste del modelo Cosinor a los perfiles de expresión de los genes *core*, cuyos valores se presentan

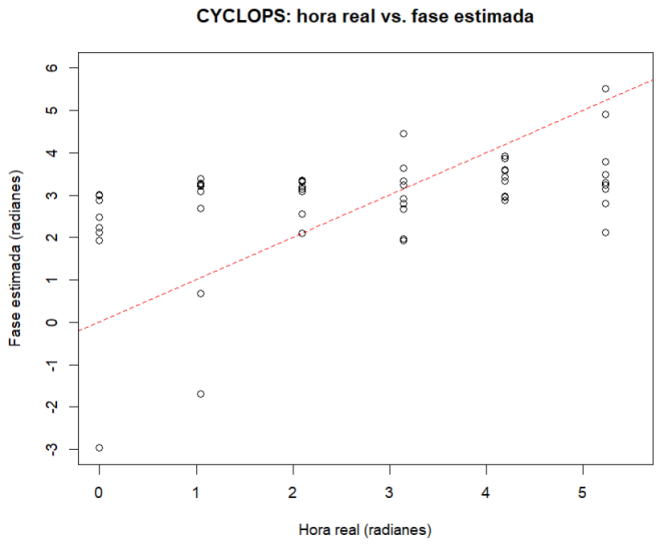

Figura 3.10: Relación entre el orden circadiano real y estimado por CYCLOPS. Se ha convertido el orden real de horas a radianes para facilitar la comparación del orden estimado.

en la Tabla 3.3. Los resultados muestran valores notablemente bajos en la mayoría de los genes, con varios por debajo de 0.1 e incluso alguno cercano a cero. Estos valores reflejan que la ordenación temporal estimada por *CYCLOPS* no consigue capturar adecuadamente la ritmicidad esperada, lo que pone en evidencia una pérdida de la estructura periódica de los datos ordenados.

Gen	R^2	**Gen**	R^2
PER1	0.089	NR1D1	0.041
PER2	0.229	RORA	0.009
PER3	0.001	STAT3	0.433
CRY1	0.134	TEF	0.115
CRY2	0.084	ARNTL	0.062
CLOCK	0.133	DBP	0.126

Tabla 3.3: Coeficiente de determinación R^2 del ajuste Cosinor para los 12 genes *core* tras la ordenación con CYCLOPS. La mayoría de los valores reflejan un ajuste deficiente.

Debido al elevado error de estimación y a la limitada capacidad para reflejar la variabilidad circadiana en los datos, CYCLOPS ha sido excluido de la segunda parte del análisis, centrada en la evaluación de los resultados obtenidos mediante modelos de aprendizaje automático. La calidad del orden temporal estimado no se considera suficientemente fiable en este conjunto de datos.

3.3. Identificación de la ritmicidad

Tras la sincronización y reordenación de la matriz de expresión, se identifican los genes con ritmicidad significativa utilizando los resultados obtenidos con CIRCUST. Para ello,

se ajusta individualmente un modelo Cosinor a cada gen de forma individual, a partir del orden estimado, lo que permite extraer los parámetros principales del ritmo y un p-valor que contrasta la hipótesis nula de ausencia de ritmicidad. Este ajuste y la obtención de dichos parámetros se han realizado mediante el test implementado en el paquete `cosinor` de R [43], descrito con más detalle en la Sección 2.3.

En total se evaluaron 12,625 genes, de los cuales 6,012 (47.6 %) se identificaron como (significativamente) rítmicos por presentar un p-valor ajustado por corrección de comparaciones múltiples inferior al umbral de significación de 0.05. Según lo descrito en la metodología, las métricas que se presentan a continuación evalúan la validez y calidad de los resultados de ritmicidad obtenidos para los 54 genes típicamente identificados como rítmicos.

Debido a que en el preprocesado de datos descrito en la Subsección 3.2.1 se descartaron 9 de los 54 genes iniciales, la comparación se lleva a cabo sobre los 45 genes restantes.

De este conjunto filtrado, se logran identificar 36 genes, lo que supone una tasa de detección del 80 %. Estos valores reflejan una elevada capacidad del algoritmo para reconstruir patrones de expresión típicamente rítmicos. Cabe destacar que dentro del conjunto de genes detectados se encuentran incluidos los 12 genes *core*, fundamentales en la regulación del reloj circadiano.

El R^2 promedio de los 36 genes identificados como rítmicos es de 0.336. Aunque este valor es inferior al esperado y también más bajo que el obtenido para el subconjunto de los 12 genes *core* ($R^2 = 0.527$), sigue indicando la presencia de un patrón oscilatorio en la expresión de dichos genes.

Este resultado sugiere que, si bien la ritmicidad en muchos de estos genes no es especialmente pronunciada, el modelo Cosinor logra capturar cierta estructura circadiana en su expresión. Estos resultados refuerzan tanto la validez del orden temporal estimado como la efectividad del procedimiento de detección de ritmicidad aplicado.

3.4. Análisis de ritmicidad con algoritmos de ML

El objetivo de esta sección es analizar la capacidad de distintos modelos supervisados de *machine learning* para aprender y reproducir los patrones de ritmicidad observados a partir de las diferentes características génicas obtenidas del resultado del método CIRCUST realizado como el Mesor, Amplitud, seno y coseno de la Acrofase (ϕ) y el coeficiente de determinación R^2. Se opta por incluir el seno y coseno de la acrofase al tratarse de una variable angular cuyo tratamiento puede condicionar los resultados de los métodos estadísticos clásicos. Además, esta decisión contribuye a una escala más homogénea entre los predictores. La Tabla 3.4 y la Tabla 3.5 recogen los estadísticos descriptivos de las variables extraídas mediante el modelo Cosinor, utilizadas como predictores en los algoritmos de aprendizaje automático. Los resultados se presentan por separado para genes clasificados como rítmicos y no rítmicos respectivamente, observándose diferencias en la distribución de la Amplitud y el R^2 para ambos grupos.

Variable	Mín.	Q1	Mediana	Media	Q3	Máx.
Mesor	-0.7096	-0.2253	0.0398	0.0390	0.2941	0.7813
Amplitud	0.0954	0.1863	0.2450	0.2733	0.3320	0.9773
$\sin(\phi)$	-1.0000	-0.8095	-0.1594	-0.0368	0.7998	1.0000
$\cos(\phi)$	-1.0000	-0.6080	-0.0896	-0.0215	0.5765	1.0000
R^2	0.0946	0.1309	0.1784	0.2174	0.2691	0.8231

Tabla 3.4: Estadísticos descriptivos de las variables predictoras del modelo Cosinor para los genes clasificados como rítmicos.

Variable	Mín.	Q1	Mediana	Media	Q3	Máx.
Mesor	-0.9525	-0.0929	0.2058	0.1641	0.4496	0.8622
Amplitud	0.0005	0.0783	0.1085	0.1100	0.1390	0.2964
$\sin(\phi)$	-1.0000	-0.5294	0.4080	0.1890	0.8743	1.0000
$\cos(\phi)$	-1.0000	-0.6617	-0.0464	-0.0248	0.6022	1.0000
R^2	0.0000	0.0229	0.0462	0.0473	0.0708	0.1045

Tabla 3.5: Estadísticos descriptivos de las variables predictoras del modelo Cosinor para los genes clasificados como no rítmicos.

Para ello, se han entrenado y comparado diversos modelos de clasificación binaria, utilizando como variable objetivo la etiqueta de ritmicidad asignada a cada gen. La calidad de las predicciones generadas por cada modelo se ha evaluado mediante métricas de rendimiento explicadas en la Sección 2.5, que permiten cuantificar su capacidad discriminativa entre ambos grupos.

Con el fin de realizar una evaluación objetiva, el conjunto de datos se ha dividido en subconjuntos de entrenamiento y *test* mediante un procedimiento estratificado. En concreto, se ha aplicado un muestreo independiente dentro de cada grupo de interés: genes rítmicos, genes no rítmicos y el subconjunto de los 54 genes principales. Se ha seleccionado aleatoriamente el 80 % de los genes de cada clase para el conjunto de entrenamiento, reservando el 20 % restante para el conjunto de *test*. De este modo, el conjunto de *test* incluye una representación equilibrada de genes principales, rítmicos y no rítmicos, lo cual permite evaluar objetivamente la capacidad de generalización de los modelos sobre tipos de genes con características distintas, incluyendo aquellos de relevancia biológica conocida que no han sido utilizados durante el entrenamiento.

3.4.1. Resultados de los algoritmos de clasificación

A continuación, se presenta para cada uno de los algoritmos estudiados en este trabajo la correspondiente matriz de confusión, junto con una evaluación cualitativa de la importancia de las variables predictoras utilizadas por cada modelo. Finalmente, en la Tabla 3.7 se compara la eficiencia de los distintos algoritmos de clasificación en términos de las cuatro métricas previamente definidas.

- **Regresión logística**: ofrece un rendimiento muy bueno con un *accuracy* del 0.988.

Presenta una sensibilidad del 0.989 y una especificidad del 0.987, lo que implica una alta capacidad para clasificar correctamente tanto genes rítmicos como no rítmicos. La exactitud balanceada (0.988) confirma el equilibrio entre clases, aunque su rendimiento es inferior al resto de modelos más complejos. Estos resultados son coherentes con la matriz de confusión obtenida tras la clasificación, donde se observa que la regresión logística es capaz de identificar correctamente como rítmicos o no rítmicos a la mayoría de los genes. En la Figura 3.11 se aprecia que el modelo logra identificar correctamente la mayoría de los genes rítmicos, con solo un pequeño número de falsos positivos y negativos.

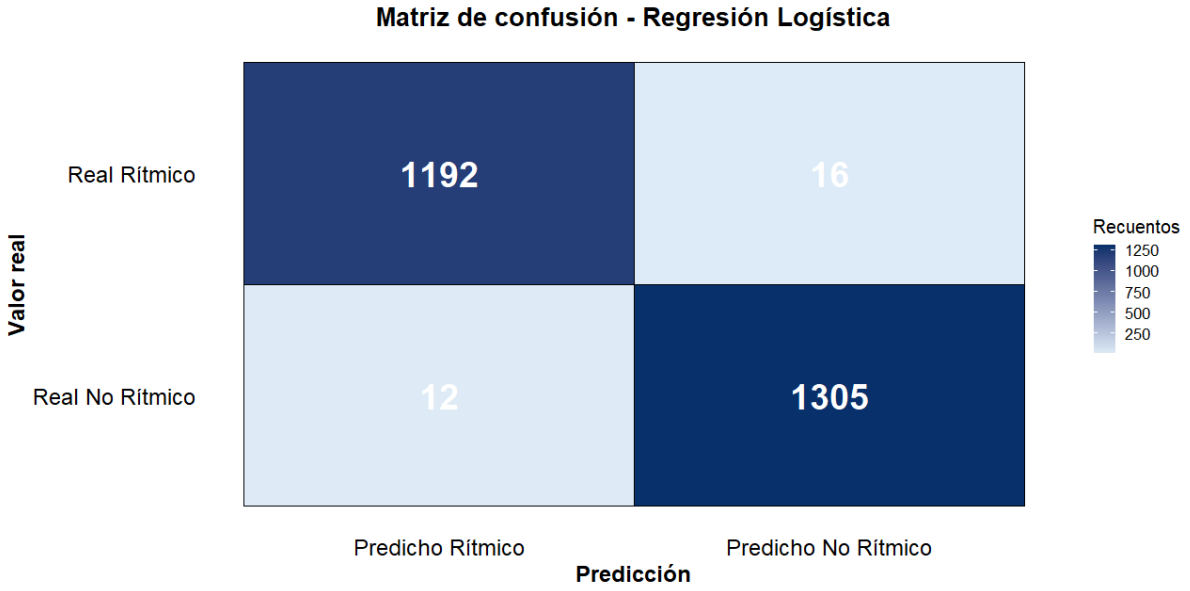

Figura 3.11: Matriz de confusión con valores absolutos para el modelo de regresión logística.

Importancia de las variables explicativas

En el modelo de regresión logística, la importancia de las variables se interpreta a partir de la magnitud y la significación estadística de sus coeficientes (ver Tabla 3.6).
El parámetro más determinante es el coeficiente de determinación R^2 del ajuste Cosinor, con un coeficiente estimado de 531.73 y un p-valor altamente significativo ($< 2 \cdot 10^{-16}$). Este resultado confirma que la calidad del ajuste sinusoidal es el factor más influyente en la predicción de ritmicidad. También se observa una contribución significativa de la Amplitud ($p = 0{,}0028$), lo que sugiere que la magnitud de la oscilación tiene un papel relevante en la clasificación.
En cambio, el Mesor y las componentes circulares de la Acrofase (seno y coseno) no resultaron significativos ($p = 0{,}4086$, $p = 0{,}3311$ y $p = 0{,}6694$, respectivamente), lo que indica que su efecto no es concluyente dentro del conjunto de predictores incluidos.

Variable	Coeficiente	Error estándar	z-valor	p-valor
Intercept	-54.373	3.395	-16.014	$<$ 2e-16
Mesor	-0.363	0.439	-0.826	0.409
Amplitud	9.209	3.081	2.989	0.0028
$\sin(\phi)$	0.189	0.195	0.972	0.331
$\cos(\phi)$	-0.075	0.175	-0.427	0.669
R^2 del Cosinor	531.729	33.149	16.041	$<$ 2e-16

Tabla 3.6: Coeficientes estimados del modelo de regresión logística y su significación estadística.

- **Random Forest**: consigue una *accuracy* del 0.995, lo que refleja un rendimiento sobresaliente. Su sensibilidad (0.993) y especificidad (0.997) indican que el modelo clasifica correctamente casi todos los genes, minimizando tanto los falsos positivos como los falsos negativos. La exactitud balanceada (0.995) evidencia un comportamiento muy equilibrado entre ambas clases. La Figura 3.12 ilustra este excelente rendimiento, mostrando predicciones prácticamente perfectas y una alta capacidad para distinguir entre genes rítmicos y no rítmicos.

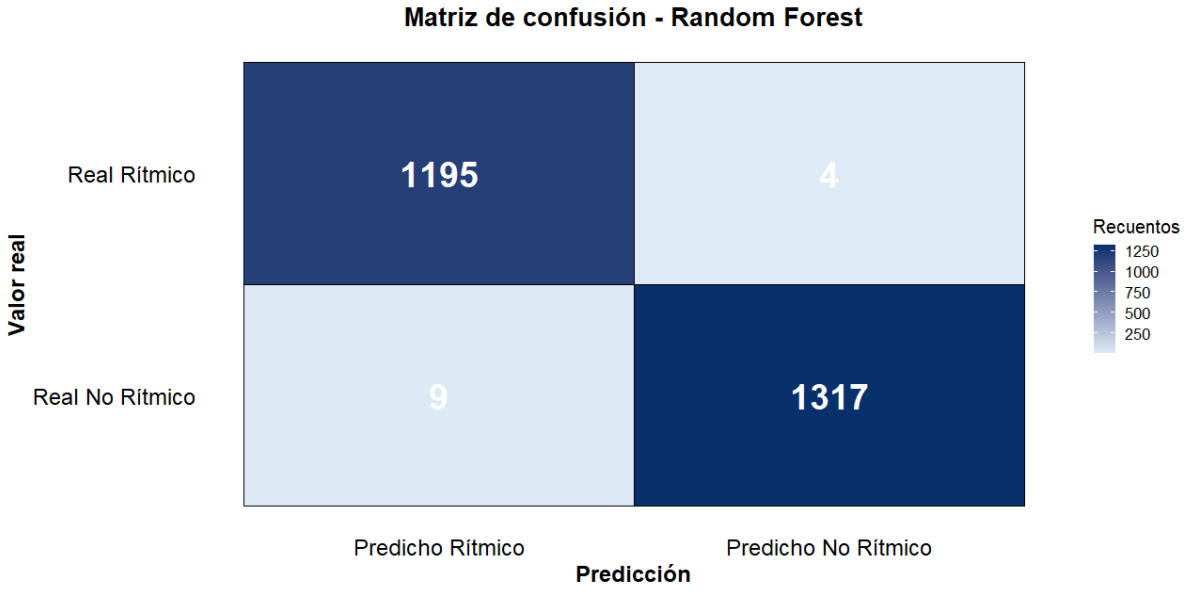

Figura 3.12: Matriz de confusión con valores absolutos para el modelo *Random Forest*.

Importancia de variables explicativas

La evaluación de la importancia de las variables en el modelo *Random Forest* se ha realizado mediante la métrica estándar de reducción de impureza acumulada basada en el índice de Gini. Los resultados confirman que el parámetro más determinante es el coeficiente de determinación R^2 del ajuste Cosinor, que alcanza un 100 % de importancia relativa. Esto indica que la regularidad y calidad del patrón oscilatorio es el principal criterio empleado por el modelo para discriminar entre genes rítmicos y no rítmicos.

Con una contribución bastante menor, pero no nula, se sitúan las componentes circulares de la Acrofase: $\cos(\phi)$ (4.25 %) y $\sin(\phi)$ (3.43 %), lo que sugiere que el momento del ciclo en que ocurre la expresión máxima tiene cierto peso en las decisiones del modelo, aunque claramente secundario frente al ajuste global.
Por el contrario, la Amplitud apenas alcanza un 1.57 % de importancia, y el Mesor no contribuye en absoluto al proceso de clasificación.

- **SVM con *kernel* radial**: obtiene una *accuracy* del 0.994, con una sensibilidad del 0.995 y una especificidad del 0.993, lo que refleja un modelo muy robusto y balanceado. Aunque presenta un rendimiento algo inferior a los modelos basados en árboles, sigue ofreciendo una clasificación muy precisa. En la Figura 3.13 se aprecia que el modelo logra plasmar la correcta clasificación de la gran mayoría de los genes, cometiendo muy pocos errores, y mantiene un equilibrio sólido entre detección y rechazo.

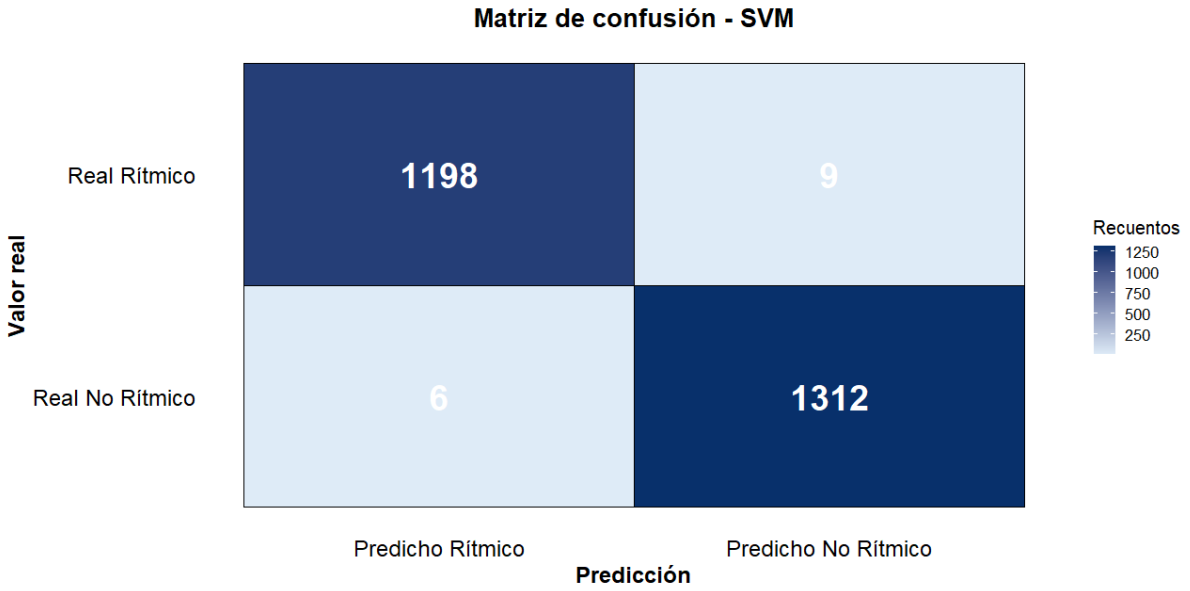

Figura 3.13: Matriz de confusión con valores absolutos para el modelo SVM con *kernel* radial.

Importancia de variables explicativas

La importancia de las variables en el modelo SVM con *kernel* radial se ha evaluado en función del impacto de cada predictor sobre el área bajo la curva ROC. Al igual que en el resto de modelos, el coeficiente de determinación R^2 del ajuste Cosinor es el atributo más relevante, con una importancia relativa del 100 %. Este resultado reafirma que la calidad del ajuste periódico es el factor clave para la clasificación entre genes rítmicos y no rítmicos.
En segundo lugar se sitúa la Amplitud (89.5 %), lo que indica que el modelo también es sensible a la magnitud de la oscilación. Le siguen el Mesor (22.7 %) y el componente seno de la Acrofase (16.3 %), que aportan información adicional sobre la estructura y el momento del ritmo circadiano.
Por el contrario, el componente coseno de la Acrofase presenta una importancia nula (0 %), lo que sugiere que no contribuye significativamente a la capacidad predictiva del

modelo en este caso.

- **XGBoost**: destaca como uno de los modelos con mejor rendimiento, alcanzando una *accuracy* de 0.995. Su sensibilidad (0.992) y especificidad (0.998) reflejan una capacidad muy equilibrada para detectar tanto genes rítmicos como no rítmicos. En conjunto, estos resultados lo posicionan como el modelo con el desempeño más alto obtenido en este análisis. La Figura 3.14 confirma estos resultados con un comportamiento muy estable del modelo, con escasos errores y una excelente capacidad de discriminación entre clases.

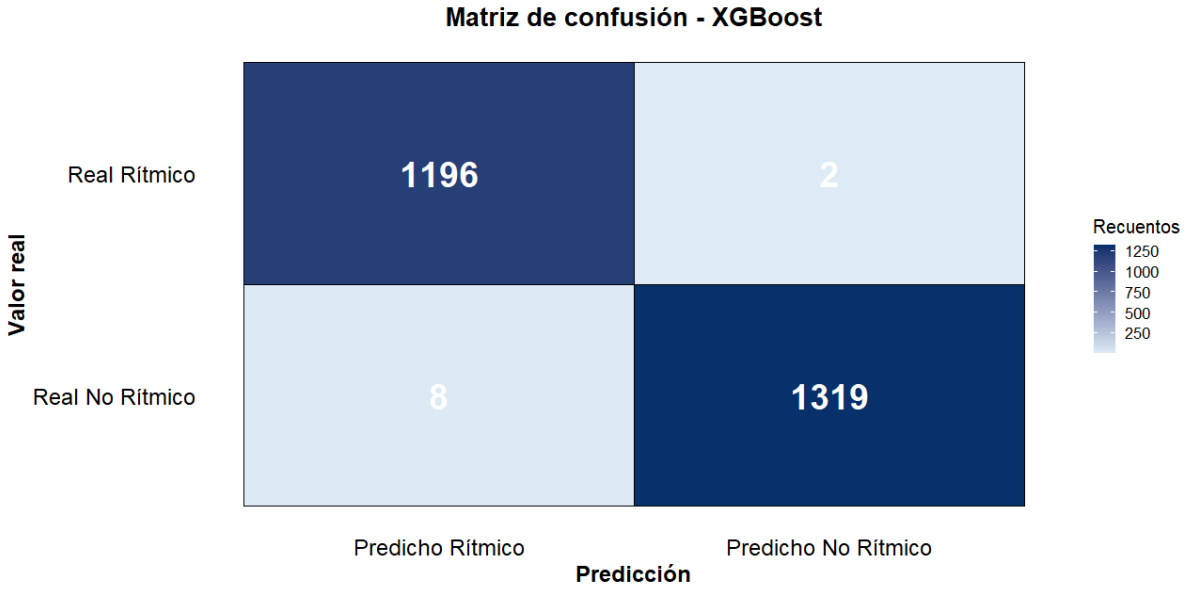

Figura 3.14: Matriz de confusión con valores absolutos para el modelo XGboost.

Importancia de variables explicativas

La importancia de las variables en el modelo *XGBoost* se ha estimado mediante el criterio `Overall`, que refleja la ganancia media de información aportada por cada predictor a lo largo de los árboles que componen el *ensemble*.
Los resultados indican que, al igual que en el resto de modelos, el coeficiente de determinación R^2 del ajuste Cosinor es con diferencia la variable más influyente, con una importancia relativa del 100 %. En segundo lugar se sitúa la Amplitud (23.5 %), que aporta información relevante sobre la magnitud de la oscilación circadiana.
Las componentes circulares de la Acrofase también contribuyen, aunque en menor medida: $\cos(\phi)$ y $\sin(\phi)$ presentan importancias del 2.11 % y 2.01 %, respectivamente. Por el contrario, el Mesor no muestra ninguna utilidad predictiva para el modelo (0 %).

De forma global, los resultados relativos al rendimiento de los modelos se resumen en la Tabla 3.7. Se observa que todos los algoritmos logran un rendimiento sobresaliente en la tarea de clasificación, con valores de exactitud superiores al 98.9 %.

Los modelos basados en árboles, *Random Forest* y *XGBoost*, destacan con una exactitud del 99.5 %, superando ligeramente al resto de métodos. En particular, *XGBoost* alcanza la mayor especificidad (0.998), lo que indica una excelente capacidad para identificar correctamente genes no rítmicos. Por su parte, el modelo *SVM* obtiene la mejor sensibilidad

(0.995), evidenciando su eficacia al detectar genes rítmicos. Aunque la regresión logística presenta un rendimiento algo inferior, sus métricas siguen siendo muy altas (*accuracy* del 98.9 %).

En conjunto, los resultados confirman la relevancia de las variables circadianas como predictores y demuestran la capacidad de estos algoritmos de ML para resolver esta tarea de forma eficaz.

Métrica	**R. Logística**	**RF**	**SVM**	**XGBoost**
Accuracy	0.989	**0.995**	0.994	**0.995**
Sensibilidad (rítmico)	0.990	0.993	**0.995**	0.992
Especificidad (no rítmico)	0.988	0.997	0.993	**0.998**
Exactitud balanceada	0.989	**0.995**	0.994	**0.995**

Tabla 3.7: Comparativa de métricas de rendimiento entre los modelos de ML utilizados para la clasificación de genes rítmicos.

Las curvas ROC obtenidas muestran un rendimiento sobresaliente en todos los modelos, con valores de AUC cercanos a 1.0 (entre 0.993 y 1.0), lo que confirma su alta capacidad para discriminar entre genes rítmicos y no rítmicos. Estos resultados reflejan que los parámetros derivados del modelo Cosinor proporcionan información suficiente para una clasificación precisa.

Capítulo 4: Conclusiones

En este trabajo se ha formulado el problema de estimación del orden temporal como un problema de optimización. Se trata de un problema fundamental para el análisis de expresión génica en humanos, especialmente en contextos donde las limitaciones éticas dificultan la obtención reiterada de muestras en el tiempo o en tejidos no accesibles de forma superficial. En particular, se han descrito y comparado dos metodologías recientes, CIRCUST y CYCLOPS, ampliamente utilizadas en la literatura que aproximan dicho problema, aplicándolas a un estudio de expresión génica en músculo esquelético humano con etiquetas temporales disponibles.

Asimismo, se ha presentado el modelo paramétrico Cosinor, referente en cronobiología para el análisis de señales oscilatorias, y su aplicación al ajuste e identificación de patrones circadianos en dicho tejido. Finalmente, a partir de los resultados obtenidos, se ha explorado el uso de algoritmos de clasificación supervisada de aprendizaje automático para la detección automática de genes rítmicos y no rítmicos, empleando como predictores los parámetros de ritmicidad estimados en el modelo Cosinor, entre los que se incluyen características clave como el Mesor, Amplitud, el seno y coseno de la Acrofase y el coeficiente de determinación R^2. Este conjunto de características captura la estructura rítmica de cada gen, facilitando el entrenamiento de modelos supervisados eficientes y generalizables.

Los resultados obtenidos en este trabajo para el problema de la estimación del orden temporal, muestran que CIRCUST ofrece un mejor rendimiento que CYCLOPS para el conjunto de datos de músculo esquelético analizado. En concreto, CIRCUST logró una estimación más precisa, como reflejan las métricas obtenidas: MAE de 1.73 horas y R^2 de 0.527, lo que indica una alta concordancia entre el orden estimado y el orden real. En contraste, CYCLOPS no logró reconstruir adecuadamente la estructura temporal latente del tejido muscular. Obtuvo resultados deficientes, con un MAE de 5.65 horas y un R^2 cercano a 0.12, reflejando una pobre capacidad de ajuste en este tipo de tejido. En consecuencia, únicamente se pudo considerar el orden estimado con CIRCUST para la identificación de ritmicidad y evaluación de los algoritmos de ML para clasificar genes rítmicos, restringiendo así el alcance de los objetivos de este trabajo.

A partir del orden estimado con la metodología CIRCUST, la metodología Cosinor permitió identificar un total de 6,012 (47.6 %) expresiones rítmicas en músculo esquelético. Posteriormente, la ritmicidad de los genes en el tejido se ha codificado como una variable binaria (rítmico/no rítmico) que actúa como variable respuesta en los cuatro modelos de clasificación supervisada evaluados en este trabajo. Las variables explicativas empleadas, como se ha mencionado, corresponden a los parámetros estimados por el modelo

Cosinor, los cuales resumen aspectos clave de la dinámica circadiana. El rendimiento de cada modelo se ha cuantificado mediante cuatro métricas, lo que ha permitido evaluar su capacidad discriminativa entre ambos grupos.

Esta estrategia presenta varias ventajas metodológicas: por un lado, permite reducir la dimensionalidad del problema y, con ello, minimizar el riesgo de sobreajuste; por otro, aporta interpretabilidad al basarse en parámetros fisiológicamente relevantes, facilitando una lectura biológicamente informada de los resultados.

Los modelos evaluados han mostrado un rendimiento sobresaliente, con áreas bajo la curva ROC superiores a 0.99 y una alta capacidad para discriminar entre genes rítmicos y no rítmicos, incluso en el caso de algoritmos lineales como la regresión logística. Entre las variables predictoras, el coeficiente de determinación (R^2) ha resultado ser el más determinante, lo que resalta el valor de la regularidad del patrón oscilatorio frente a otras características como la Amplitud, la Acrofase o el pico de máxima expresión.

En conjunto, estos resultados confirman que los parámetros derivados del modelo Cosinor capturan información suficiente y relevante para una clasificación precisa, consolidando esta aproximación como una herramienta sólida y prometedora para el análisis de ritmicidad circadiana en estudios transcriptómicos humanos.

4.1. Propuestas de investigación futura

A partir de este trabajo, se proponen las siguientes líneas de investigación futura:

- Evaluación del rendimiento de CYCLOPS en tejido de músculo esquelético. Se propone investigar las limitaciones del algoritmo en este tipo de tejido, caracterizado por una señal circadiana débil. En particular, se estudiará por qué CYCLOPS no logra estimar correctamente el orden temporal.
- Modelización e identificación de ritmicidad mediante otros modelos paramétricos como el modelo FMM [15], de particular interés para el análisis de expresiones oscilatorias asimétricas.
- Incorporación de herramientas de interpretabilidad en los modelos de aprendizaje automático, como los valores SHAP (SHapley Additive exPlanations), basados en teoría de juegos [44]. Estas técnicas permiten atribuir de manera equitativa la contribución de cada variable a una predicción individual, facilitando una comprensión más profunda del comportamiento del modelo y de los factores que determinan la clasificación de los genes como rítmicos o no rítmicos.
- Extensión y validación de la metodología en bases de datos de expresión génica humana correspondientes a tejidos con menor ritmicidad que el músculo esquelético, utilizando conjuntos de datos con etiquetas temporales reales o muestras *postmortem* con información sobre la hora de la muerte. Esta aplicación permitiría realizar una evaluación más robusta y realista del desempeño de los algoritmos de clasificación, especialmente en contextos con mayor variabilidad interindividual y señales circadianas menos marcadas.

Bibliografía

[1] Imagen del Proceso de Expresión Génica del Gen. `https://es.khanacademy.org/science/ap-biology/gene-expression-and-regulation/transcription-and-rna-processing/a/overview-of-transcription`. Acceso: 2025-03-25.

[2] David P. Clark and Nanette J. Pazdernik. *Molecular Biology*. Elsevier, 2012. ISBN: 9780123785947.

[3] Yolanda Larriba, Ivy C. Mason, Richa Saxena, Frank A. J. L. Scheer, and Cristina Rueda. CIRCUST: A Novel Methodology for Temporal Order Reconstruction of Molecular Rhythms; Validation and Application Towards a Daily Rhythm Gene Expression Atlas in Humans. *PLOS Computational Biology*, 19(9):e1011510, 2023.

[4] D. C. Klein, R. Y. Moore, and S. M. Reppert, editors. *Suprachiasmatic Nucleus: The Mind's Clock*. Oxford University Press, New York, 1991.

[5] Ray Zhang, Nicholas F. Lahens, Heather I. Ballance, Michael E. Hughes, and John B. Hogenesch. A Circadian Gene Expression Atlas in Mammals: Implications for Biology and Medicine. *Proceedings of the National Academy of Sciences*, 111(45):16219–16224, 2014.

[6] Michael E. Hughes, Luciano DiTacchio, Kevin R. Hayes, Christopher Vollmers, Sridhar Pulivarthy, Julie E. Baggs, Satchidananda Panda, and John B. Hogenesch. Harmonics of Circadian Gene Transcription in Mammals. *PLoS Genetics*, 5(4):e1000442, 2009.

[7] Francisco Javier Martínez Picabea. Crono-Oncología: El Ciclo Circadiano Como Nuevo Abordaje en el Diagnóstico de Cáncer. Trabajo Fin de Grado. Grado de Biotecnología, 2021. Universidad Politécnica de Madrid.

[8] Gabriele Sulli, Emily N. C. Manoogian, Pam R. Taub, and Satchidananda Panda. Training the Circadian Clock, Clocking the Drugs, and Drugging the Clock to Prevent, Manage, and Treat Chronic Diseases. *Trends in Pharmacological Sciences*, 39(9):812–827, 2018.

[9] Saurabh Sahar and Paolo Sassone-Corsi. Metabolism and Cancer: The Circadian Clock Connection. *Nature Reviews Cancer*, 9(12):886–896, 2009.

[10] Daniel P. Cardinali, Gregory M. Brown, and Seithikurippu R. Pandi-Perumal. Chronotherapy. In *Handbook of Clinical Neurology*, volume 179, pages 357–370. Elsevier, 2021.

[11] Steven A. Brown, Fabienne Fleury-Olela, Emi Nagoshi, Conrad Hauser, Cristiana Juge, Christophe A. Meier, Rachel Chicheportiche, Jean-Michel Dayer, Urs Albrecht, and Ueli Schibler. The Period Length of Fibroblast Circadian Gene Expression Varies Widely Among Human Individuals. *PLoS Biology*, 3(10):e338, 2005.

[12] Tracy A. Bedrosian and R. J. Nelson. Timing of Light Exposure Affects Mood and Brain Circuits. *Translational Psychiatry*, 7(1):e1017, 2017.

[13] William H. Walker, James C. Walton, A. Courtney DeVries, and Randy J. Nelson. Circadian rhythm disruption and mental health. *Translational Psychiatry*, 10(1):28, 2020.

[14] Yizhang Zhu, Likun Wang, Yuxin Yin, and Ence Yang. Systematic Analysis of Gene Expression Patterns Associated with Postmortem Interval in Human Tissues. *Scientific Reports*, 7(1):5435, 2017.

[15] Cristina Rueda, Yolanda Larriba, and Shyamal D. Peddada. Frequency Modulated Möbius Model Accurately Predicts Rhythmic Signals in Biological and Physical Sciences. *Scientific Reports*, 9(1):18701, 2019.

[16] Jie Li, Pengxing Nie, Christoph W. Turck, and Guang-Zhong Wang. Gene Networks Under Circadian Control Exhibit Diurnal Organization in Primate Organs. *Communications Biology*, 5(1):764, 2022.

[17] Cho-Yi Chen, Ryan W. Logan, Tianzhou Ma, David A. Lewis, George C. Tseng, Etienne Sibille, and Colleen A. McClung. Effects of Aging on Circadian Patterns of Gene Expression in the Human Prefrontal Cortex. *Proceedings of the National Academy of Sciences*, 113(1):206–211, 2016.

[18] Katrin Ackermann. *Analyses of Diurnal Rhythms in Human Post-Mortem Tissues*. PhD thesis, Universität Frankfurt am Main, 2007. Dissertation.

[19] Germaine Cornelissen. Cosinor-Based Rhythmometry. *Theoretical Biology and Medical Modelling*, 11:1–24, 2014.

[20] Ron C. Anafi, Lauren J. Francey, John B. Hogenesch, and Junhyong Kim. CYCLOPS Reveals Human Transcriptional Rhythms in Health and Disease. *Proceedings of the National Academy of Sciences*, 114(20):5312–5317, 2017.

[21] Yolanda Larriba, Cristina Rueda, Miguel A. Fernández, and Shyamal D. Peddada. Order Restricted Inference for Oscillatory Systems for Detecting Rhythmic Signals. *Nucleic Acids Research*, 44(22):e163–e163, 2016.

[22] Gang Wu, Marc D. Ruben, Robert E. Schmidt, Lauren J. Francey, David F. Smith, Ron C. Anafi, Jacob J. Hughey, Ryan Tasseff, Joseph D. Sherrill, John E. Oblong, et al. Population-Level Rhythms in Human Skin with Implications for Circadian Medicine. *Proceedings of the National Academy of Sciences*, 115(48):12313–12318, 2018.

[23] Marc D. Ruben, Gang Wu, David F. Smith, Robert E. Schmidt, Lauren J. Francey, Yin Yeng Lee, Ron C. Anafi, and John B. Hogenesch. A Database of Tissue-Specific Rhythmically Expressed Human Genes Has Potential Applications in Circadian Medicine. *Science Translational Medicine*, 10(458):eaat8806, 2018.

[24] Hiroki R. Ueda, Wenbin Chen, Yoichi Minami, Sato Honma, Kenichi Honma, Masamitsu Iino, and Seiichi Hashimoto. Molecular-Timetable Methods for Detection of Body Time and Rhythm Disorders from Single-Time-Point Genome-Wide Expression Profiles. *Proceedings of the National Academy of Sciences*, 101(31):11227–11232, 2004.

[25] Forest Agostinelli, Nicholas Ceglia, Babak Shahbaba, Paolo Sassone-Corsi, and Pierre Baldi. What Time Is It? Deep Learning Approaches for Circadian Rhythms. *Bioinformatics*, 32(12):i8–i17, 2016.

[26] Rosemary Braun, William L. Kath, Marta Iwanaszko, Elzbieta Kula-Eversole, Sabra M. Abbott, Kathryn J. Reid, Phyllis C. Zee, and Ravi Allada. Universal Method for Robust Detection of Circadian State from Gene Expression. *Proceedings of the National Academy of Sciences*, 115(39):E9247–E9256, 2018.

[27] Michael E. Hughes, John B. Hogenesch, and Karl Kornacker. JTK_CYCLE: An Efficient Nonparametric Algorithm for Detecting Rhythmic Components in Genome-Scale Data Sets. *Journal of Biological Rhythms*, 25(5):372–380, 2010.

[28] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[29] David W. Hosmer Jr., Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, 2013.

[30] Kaitlin Kirasich, Trace Smith, and Bivin Sadler. Random Forest vs Logistic Regression: Binary Classification for Heterogeneous Datasets. *SMU Data Science Review*, 1(3):9, 2018.

[31] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[32] David Legorreta. Entradas sobre *Support Vector Machines* (SVM) en el blog *dlegorreta*. `https://dlegorreta.wordpress.com/tag/svm/`, 2025. Consultado el 3 de junio de 2025.

[33] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[34] Martin Thoma - Wikimedia Commons. ROC Curve. `https://commons.wikimedia.org/wiki/File:Roc_curve.svg`. Consultado el 5 de junio de 2025.

[35] Laurent Perrin, Ursula Loizides-Mangold, Stéphanie Chanon, Cédric Gobet, Nicolas Hulo, Laura Isenegger, Benjamin D. Weger, Eugenia Migliavacca, Aline Charpagne, James A. Betts, et al. Transcriptomic Analyses Reveal Rhythmic and CLOCK-Driven Pathways in Human Skeletal Muscle. *eLife*, 7:e34114, 2018.

[36] Ursula Loizides-Mangold, Laurent Perrin, Bart Vandereycken, James A. Betts, Jean-Philippe Walhin, Iain Templeman, Stéphanie Chanon, Benjamin D. Weger, Christine

Durand, Maud Robert, et al. Lipidomics Reveals Diurnal Lipid Oscillations in Human Skeletal Muscle Persisting in Cellular Myotubes Cultured in Vitro. *Proceedings of the National Academy of Sciences*, 114(41):E8565–E8574, 2017.

[37] Lorenzo Talamanca and Felix Naef. How to Tell Time: Advances in Decoding Circadian Phase from Omics Snapshots. *F1000Research*, 9, 2020.

[38] Aram Ansary Ogholbake and Qiang Cheng. Gene Expression Clock: An Unsupervised Deep Learning Approach for Predicting Circadian Rhythmicity from Whole Genome Expression. *Neural Computing and Applications*, 36(33):20653–20670, 2024.

[39] Yolanda Larriba. CIRCUST algorithm. `https://github.com/yolandalago/CIRCUST`, 2023. Último acceso: 9 de julio de 2025.

[40] Ron Anafi. CYCLOPS: Circadian Clock Estimation from Expression Data. `https://github.com/ranafi/CYCLOPS`, 2017. Accedido el 4 de junio de 2025.

[41] Neel Rafi and Ron Anafi. CYCLOPS-2.0: Cyclic Ordering by Periodic Structure. `https://github.com/ranafi/CYCLOPS-2.0`, 2017. Último acceso: 4 de junio de 2025.

[42] Lorenzo Talamanca. *Statistical Physics of Periodic Biological Processes*. PhD thesis, EPFL, 2023.

[43] Adrian Barnett. Introduction to the `cosinor` Package. `https://cran.r-project.org/web/packages/cosinor/vignettes/Intro.html`, 2020. Último acceso: 4 de junio de 2025.

[44] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[45] Anirban DasGupta. *The Trimmed Mean*, pages 271–278. Springer New York, New York, NY, 2008.

[46] Hervé Abdi and Lynne J. Williams. Principal Component Analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[47] Michael J Kirby and Rick Miranda. Circular Nodes in Neural Networks. *Neural Computation*, 8(2):390–402, 1996.

[48] Matthias Scholz. Analysing Periodic Phenomena by Circular PCA. In *International conference on bioinformatics research and development*, pages 38–47. Springer, 2007.

Apéndice A: Metodología adicional

A.1. Media recortada

La media recortada es una medida robusta de tendencia central que elimina una proporción de los valores extremos antes de calcular la media [45]. Para una variable $X_i \in \mathbb{R}^m$, con m observaciones ordenadas de forma creciente, la media recortada se define como:

$$\bar{X}_i = \frac{1}{[m - m \times \alpha]} \sum_{j=1}^{[m-m\times\alpha]} x_{i,(j)}$$

donde:

- $x_{i,(j)}$ denota el j-ésimo valor ordenado del vector X_i.
- $\alpha \in [0, 1)$ es la proporción de recorte aplicada (por ejemplo, $\alpha = 0{,}1$ para un 10 %).
- $[m - m \times \alpha]$ representa la parte entera del número de observaciones consideradas tras el recorte.

A.2. Residuos estandarizados

Esta magnitud mide la diferencia entre el valor observado y el valor ajustado por el modelo, normalizada por la variabilidad residual del ajuste. Se define el residuo estandarizado r_j^* de cada muestra j (para un gen fijo) como:

$$r_j^* = \frac{|x_{ij} - \hat{x}_{ij}|}{\sqrt{\text{SSE}_i/(m-3)}}$$

donde:

- x_{ij} es el valor observado del gen i en la muestra j.
- $\hat{x}_{ij}$ es el valor ajustado por el modelo Cosinor para el mismo gen y muestra.
- SSE_i representa la suma de cuadrados del error del ajuste del gen i, calculada como:

$$\text{SSE}_i = \sum_{j=1}^{m} (x_{ij} - \hat{x}_{ij})^2$$

- m es el número total de muestras.
- El denominador $(m-3)$ corresponde a los grados de libertad del modelo Cosinor, que estima tres parámetros: mesor, amplitud y fase.

A.3. Contraste de ritmicidad del modelo Cosinor

Para evaluar si un gen presenta una oscilación circadiana significativa se plantea un contraste de hipótesis sobre la amplitud del modelo Cosinor. Partiendo de la formulación no lineal, se aplica la siguiente reparametrización lineal [19]:

$$x(t_j) = M + \beta\cos(\omega t_j) + \gamma\sin(\omega t_j),$$

donde

$$\omega = \frac{2\pi}{T}, \qquad T = 24\text{ h}, \qquad \beta = A\cos\phi, \qquad \gamma = -A\sin\phi,$$

obteniendo

$$A = \sqrt{\beta^2+\gamma^2} \quad \text{(amplitud)}, \qquad \phi = \arctan\left(\frac{-\gamma}{\beta}\right) \text{(fase)}.$$

El contraste se formula como

$$\begin{aligned} H_0 &: \beta = \gamma = 0 \qquad \text{(sin componente rítmica)} \\ H_1 &: (\beta,\gamma) \neq (0,0) \quad \text{(ritmo circadiano presente)}. \end{aligned}$$

Estadístico *F*. Sea SSE_R la suma de cuadrados del error del modelo reducido (solo mesor) y SSE_F la del modelo completo (incluye los dos términos trigonométricos). Con m observaciones:

$$F = \frac{(\text{SSE}_R - \text{SSE}_F)/2}{\text{SSE}_F/(m-3)} \;\sim\; F_{(2,\,m-3)}.$$

Se rechaza H_0 y se concluye que el gen es rítmico si $F > F_{2,m-3;\,\alpha}$, con α nivel de significación.

A.4. Descomposición en Valores Singulares (SVD, Singular Value Decomposition)

La Descomposición en Valores Singulares es una técnica de factorización matricial que descompone una matriz real o compleja (X) de dimensión $m \times n$ en el producto de

tres matrices: una matriz unitaria (U) de dimensión $m \times m$, una matriz diagonal Σ de dimensión $m \times n$ con entradas no negativas en la diagonal (conocidos como valores singulares), y la transpuesta conjugada de una matriz unitaria (V^T) de dimensión $n \times n$. Matemáticamente, se expresa como:

$$X = U\Sigma V^T \tag{A.1}$$

En el contexto del análisis de expresión génica, la matriz $X_{m \times n}$ puede representar las m muestras diferentes en n niveles de expresión de genes. La SVD se utiliza para identificar los patrones principales de variación en los datos. Las columnas de U representan combinaciones ortonormales de las muestras originales, las filas de V^T representan combinaciones de genes, y los valores singulares en Σ indican la importancia relativa o la cantidad de varianza explicada por cada componente.

Los primeros vectores singulares, correspondientes a los *eigenvalues* más grandes, capturan la mayor parte de la varianza en los datos y pueden interpretarse como *eigengenes* que corresponden a patrones de expresión fuertes y cíclicos. Además, los *eigenvalues* más pequeños se descartarían, ya que reduciríamos la cantidad de datos no relevantes, eliminando ruido en nuestro estudio.

Si se retienen los dos primeros valores singulares, puede proyectarse cada muestra $x_i \in R^m$ en un plano con:

$$x_i' = x_i V_2 \tag{A.2}$$

donde $V_2 \in R^{m \times 2}$ contiene los dos primeros vectores singulares. Esta proyección es útil para visualizar agrupamientos (por ejemplo, la fase circadiana).

A.5. Análisis de Componentes Principales (PCA)

El Análisis de Componentes Principales [46] es una técnica lineal y no supervisada que se utiliza para reducir la dimensionalidad de un conjunto de datos, transformándolo en un nuevo conjunto de variables, llamadas componentes principales, que son combinaciones lineales ortogonales de las variables originales. El primer componente principal captura la mayor varianza posible de los datos, el segundo componente principal captura la segunda mayor varianza posible, siendo ortogonal al primero, y así sucesivamente. El objetivo es encontrar una nueva base de coordenadas donde los datos sean más fáciles de analizar.

El PCA es ampliamente utilizado en expresión génica para reducir el ruido, visualizar la estructura de los datos y extraer las principales fuentes de variación.

Si tenemos una matriz de datos X de dimensión $m \times n$, donde m es el número de genes y n es el número de muestras. PCA sigue los siguientes pasos:

- 1.Centrar los datos: restar la media de cada variable a sus respectivos valores.

$$X_c = X - \bar{X}$$

Donde $\bar{X}$ es el vector de medias de las columnas, definido como:

$$\bar{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$$

- 2. Calcular la matriz de covarianza: comprobar la variabilidad conjunta entre las variables. La matriz de covarianza C se define como:

$$C = \frac{1}{m-1} X_c X_c^T$$

- 3. Calcular los autovectores y autovalores de la matriz de covarianza: Los autovectores representan las direcciones de los componentes principales, y los autovalores indican la cantidad de varianza explicada por cada componente. Los autovalores λ se obtienen resolviendo la ecuación característica:

$$\det(C - \lambda I) = 0$$

Donde I es la matriz identidad de tamaño $n \times n$.

- 4. Seleccionar los componentes principales: Elegir los k autovectores correspondientes a los k autovalores más grandes para retener la mayor parte de la varianza. Se ordenan los autovalores en orden descendente:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

y se seleccionan los primeros k autovectores correspondientes a los k autovalores más grandes en base a diferentes técnicas.
Entre las más conocidas encontramos:
-Seleccionar los autovalores que superen un umbral de varianza explicada acumulada, normalmente 90 %.
-El criterio del codo (gráfico), mantener los autovalores que se encuentran antes de formar el codo.
-La regla de Kaiser, que consiste en quedarse con los autovalores mayores que 1.

- 5. Proyectar los datos: Transformar los datos originales en el nuevo espacio de menor dimensión formado por los k componentes principales seleccionados. La proyección de los datos en la nueva base de componentes principales se obtiene como:

$$Z = X_c V$$

Donde V es la matriz de autovectores seleccionados y X_c la matriz con los datos centrados.

A.6. Autoencoder Circular

Los *autoencoders* son redes neuronales no supervisadas diseñadas para aprender representaciones comprimidas de los datos de entrada. En su forma básica, constan de un

codificador que transforma un vector de entrada $x \in R^n$ en una representación latente $z \in R^k$, y un decodificador que intenta reconstruir $\hat{x}$ desde z, minimizando el error de reconstrucción $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. Esta arquitectura fuerza a la red a capturar las características esenciales de los datos.

Restricción circular

El autoencoder circular propuesto por Kirby y Miranda [47,48] extiende esta arquitectura al caso de variables cíclicas. En particular, se diseña una capa de cuello de botella con dos neuronas acopladas que restringen la codificación a la circunferencia unidad.

En este contexto, cada muestra $j = 1, \ldots, m$ está representada por un vector de expresión génica $\mathbf{x}^{(j)} = (x_{1j}, x_{2j}, \ldots, x_{nj})$. El objetivo es mapear dicha muestra a un punto sobre la circunferencia unidad, interpretando su posición angular como una estimación de fase temporal $\theta_j \in [0, 2\pi)$.

Codificación circular

Como se muestra en la Figura A.1, el autoencoder circular introduce una capa latente circular (*bottleneck layer*), compuesta por dos neuronas p y q, cuyos valores de salida están restringidos a la circunferencia unidad mediante la relación: $z_p^2 + z_q^2 = 1$.

Esto permite representar la codificación latente de una muestra mediante una única variable angular θ, donde $z_p = cos(\theta), z_q = sin(\theta)$.

Funcionamiento de la red

Como en una red neuronal tradicional, ambas neuronas en esta capa latente, calculan una suma ponderada (con sus respectivos pesos w_{ps}, w_{qs}) de las activaciones z_s dadas de las s neuronas de la capa anterior.

$$o_p = \sum_s w_{ps} z_s \qquad\qquad o_q = \sum_s w_{qs} z_s \tag{A.3}$$

Las sumas o_p y o_q se normalizan para poder proyectarse en el círculo unidad:

$$z_p = \frac{o_p}{\sqrt{o_p^2 + o_q^2}} \qquad z_q = \frac{o_q}{\sqrt{o_p^2 + o_q^2}} \tag{A.4}$$

Finalmente, se calcula la fase angular asociada a la muestra:

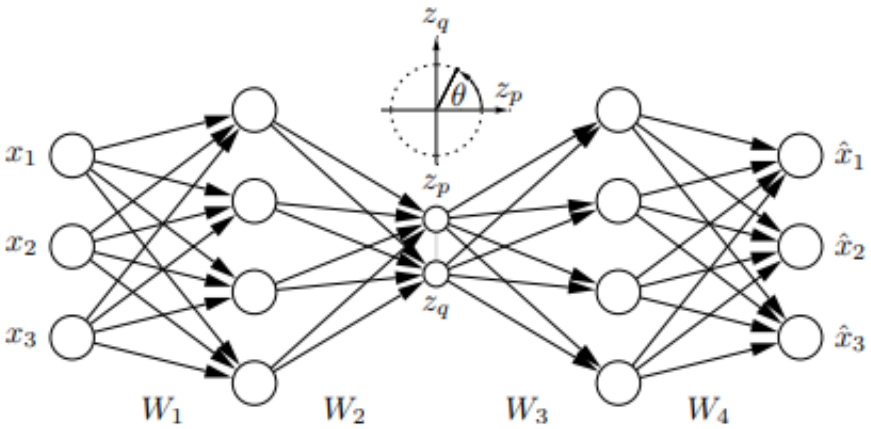

Figura A.1: Arquitectura de una red circular PCA (CPCA). Se observan las neuronas circulares (p,q) de la capa latente. Los valores z_p y z_q se restringen al círculo unidad, por lo que se pueden representar mediante una única variable angular θ. Imagen obtenida de [48].

$$\theta_j = \arctan\left(\frac{z_q}{z_p}\right) \tag{A.5}$$

De este modo, cada muestra se codifica en un único valor angular que representa su localización en un ciclo periódico completo.

A.7. Outliers detectados en CIRCUST

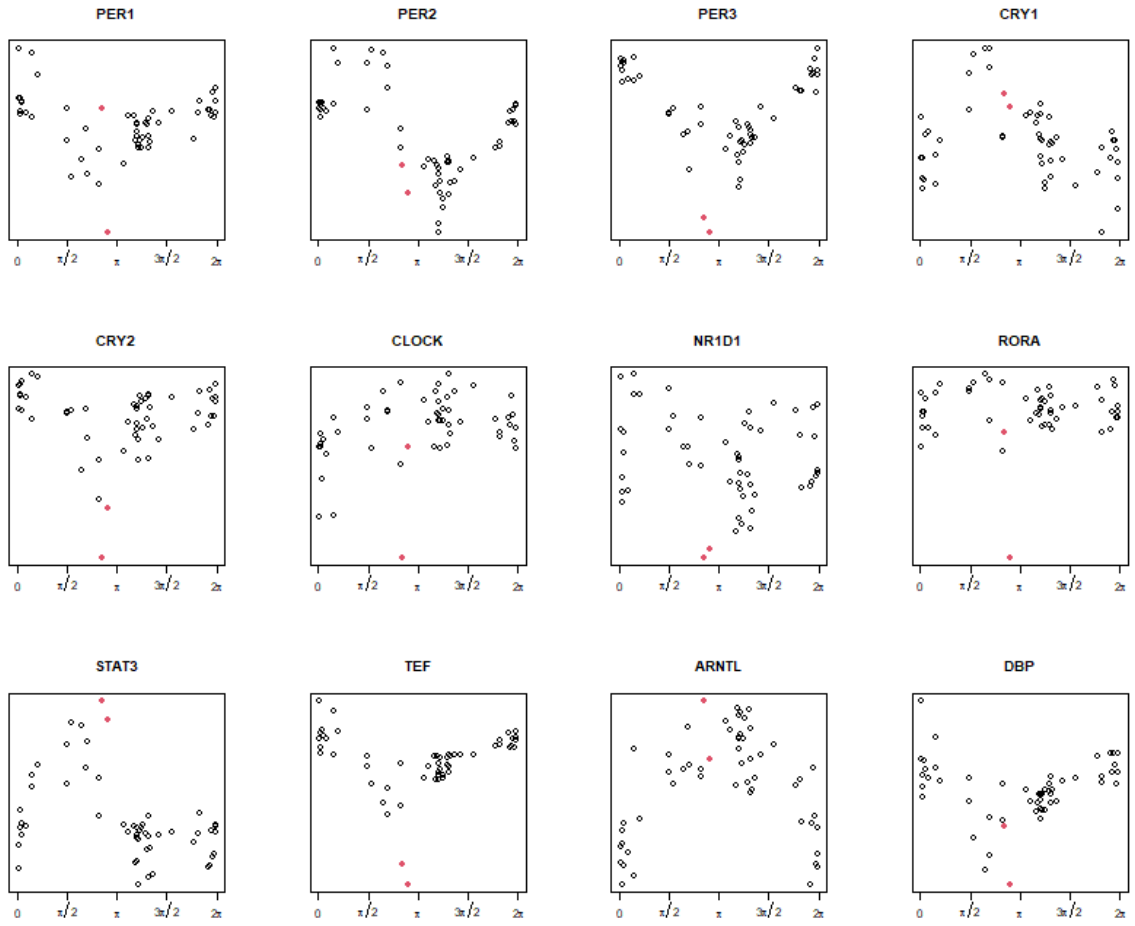

Figura A.2: Representación de la expresión ordenada de los genes *core* donde se marca en rojo los *outliers* detectados.

A.8. Elección de hiperparámetros en los modelos de aprendizaje supervisado

Regresión logística

La regresión logística, al ser un modelo relativamente sencillo y lineal, requiere un número reducido de hiperparámetros. En este trabajo se ha optado por su versión sin regularización, es decir, sin aplicar penalizaciones sobre los coeficientes. Esta elección permite evaluar su capacidad predictiva de forma directa, sin introducir sesgos adicionales debidos a técnicas de ajuste o reducción de complejidad.

El objetivo es utilizarla como modelo base o de referencia, lo que justifica un enfoque minimalista en su configuración. Asimismo, se ha mantenido el umbral de decisión en

su valor por defecto (τ=0.5), alineado con la interpretación probabilística estándar del modelo, en la que se predice la clase positiva si la probabilidad estimada supera el 50 %.

Random Forest

En esta implementación, se ha utilizado la configuración por defecto del algoritmo, sin ajuste explícito de hiperparámetros como el número de árboles o la cantidad de predictores seleccionados aleatoriamente en cada división.

En particular, el número de árboles (`ntree`) se ha fijado automáticamente en 500, un valor habitual que proporciona un buen equilibrio entre estabilidad y coste computacional. Respecto al número de variables evaluadas en cada partición (`mtry`), el valor empleado es $\sqrt{p}$, siendo p el número total de predictores. Este valor promueve la diversidad entre árboles del *ensemble*, lo que resulta fundamental para mejorar la generalización del modelo. El modelo ha utilizado la métrica de impureza basada en el índice de Gini para construir las particiones, y el muestreo con reemplazo (*bootstrap*), lo que permite estimar el error fuera de bolsa (*out-of-bag*, OOB), aunque en este caso la evaluación se ha realizado mediante validación cruzada.

SVM

El modelo SVM utilizado en este trabajo se ha entrenado empleando un *kernel* de base radial y ajustando automáticamente sus hiperparámetros mediante búsqueda en rejilla con `tuneLength = 5`. Además, se aplicó preprocesamiento previo de los datos con centrado y escalado. Dos de los hiperparámetros fundamentales en este modelo son el parámetro de penalización C y el parámetro γ asociado al *kernel* radial. Ambos hiperparámetros se han ajustado automáticamente mediante validación cruzada interna, evaluando distintas combinaciones. Esta estrategia permite identificar la configuración óptima sin riesgo de sobreajuste al conjunto de test. En este caso, no se aplicó penalización explícita por desbalance de clases, ya que el modelo mostró un rendimiento equilibrado.

XGBoost

Se han mantenido los valores por defecto para los principales hiperparámetros de *XGBoost*, al considerar que ofrecen un equilibrio razonable entre rendimiento y simplicidad interpretativa en un primer análisis comparativo. En concreto, se ha utilizado un número total de iteraciones igual a 100, una tasa de aprendizaje moderada de 0.3, y una profundidad máxima de los árboles de 6 niveles, lo que permite capturar interacciones relevantes sin inducir un sobreajuste excesivo.

Asimismo, se ha utilizado el 100 % de las observaciones y de las variables en cada árbol individual (`subsample` = 1 y `colsample_bytree` = 1), por lo que no se ha aplicado submuestreo en esta configuración inicial. En cuanto a los términos de regularización, también se han mantenido sus valores predeterminados: `reg_lambda` = 1, correspondiente a la penalización L_2 y `reg_alpha` = 0, asociada a la penalización L_1. Esta configuración

implica que se aplica una penalización cuadrática moderada sobre los pesos de las hojas para controlar la complejidad del modelo.

Apéndice B: Código desarrollado

Este apéndice recoge el conjunto de scripts y fragmentos de código desarrollados para llevar a cabo el análisis completo presentado en este trabajo. Todas las implementaciones han sido realizadas en el lenguaje R, utilizando diversas librerías especializadas en procesamiento de datos, modelado estadístico y visualización gráfica.

El código incluido se encuentra estructurado por secciones, siguiendo el mismo orden lógico del cuerpo principal del documento: desde el preprocesamiento y ordenación temporal de las muestras hasta el ajuste de modelos Cosinor, la detección de ritmicidad y la evaluación mediante algoritmos de ML.

B.1. Funciones auxiliares

```r
# Funcion para obtener genes con m s del 30% de valores iguales a cero
obtener_genes_con_muchos_ceros <- function(data, umbral = 0.3) {
  n_muestras <- ncol(data)
  genes_filtrados <- data[rowSums(data == 0) > (umbral * n_muestras), ]
  return(genes_filtrados)
}

#Funcion para filtrar genes segun la media recortada
filtrarGenesPorMCut <- function(data, porcentaje = 0.9, threshold =
    0.4) {
  cut <- floor(porcentaje * ncol(data))
  addMeanCut <- numeric(nrow(data))
  addRow <- logical(nrow(data))

  for (i in 1:nrow(data)) {
    expr_values <- as.numeric(data[i, ])
    mCut <- mean(sort(expr_values)[1:cut])
    addMeanCut[i] <- mCut
    addRow[i] <- abs(mCut) > threshold
  }

  # Subconjunto de genes que pasan el umbral
  data_filtrada <- data[addRow, ]

  # Tambien se devuelve el vector de mCuts por si se quiere
      inspeccionar
  return(list(data_filtrada = data_filtrada, mCut_values = addMeanCut))
}

```

```r
#Funcion que devuelve la proporcion de genes que superan el valor del threshold
test_threshold <- function(threshold, mCut_values, gene_names, core_genes) {
  # Genes retenidos
  selected <- gene_names[abs(mCut_values) > threshold]

  prop_retained <- length(selected) / length(gene_names)
  prop_core <- sum(core_genes %in% selected) / length(core_genes)

  return(c(prop_retained, prop_core))
}

# Funcion que normaliza un vector entre [-1,1] utilizado min-max
normalize <- function(x) {
  return((2 * (x - min(x)) / (max(x) - min(x))) - 1)
}

# Funcion que proyecta E1 y E2 sobre el circulo unidad
CPCA_order <- function(E1, E2) {
  e1 <- E1 / sqrt(E1^2 + E2^2)
  e2 <- E2 / sqrt(E1^2 + E2^2)
  return(atan2(e2, e1))
}

# FuncionCosinor debe devolver los valores ajustados como primer elemento
CosinorOutliers <- function(data, time, periodo = 2 * pi) {
  # Ajustar modelo Cosinor
  fit <- funcionCosinor(data, time, periodo)
  fitted_vals <- fit[[1]]

  # Calcular residuos y residuos studentizados
  residuales <- data - fitted_vals
  sse <- sum(residuales^2)
  std_resid <- abs(residuales) / sqrt(sse / (length(data) - 3))

  # Retorna un vector l gico: TRUE si no es outlier, FALSE si es
  return(std_resid < 3)
}

# Funcion que ajusta el modelo Cosinor
funcionCosinor <- function(datos, time, periodo) {
  xx <- cos(time)
  zz <- sin(time)

  fit <- lm((datos) ~ xx + zz)
  Mest <- fit$coefficients[1]
  bb <- fit$coefficients[2]
  gg <- fit$coefficients[3]
  phiEst <- atan2(-gg, bb) %% (2 * pi)
  Aest <- sqrt(bb^2 + gg^2)

  y_fit <- Mest + Aest * cos(time + phiEst)
  r2 <- summary(fit)$r.squared  # A adir R

  return(list(y_fit, Mest, Aest, (time + phiEst) %% (2 * pi), phiEst, r2))
```

```r
}

#Funcion que devuelve la curva Cosinor con gran precision
generateCosinor <- function(M, A, phi, length.out = 200, periodo = 2 *
    pi, plot = FALSE) {
  # Tiempo en radianes
  t <- seq(0, periodo, length.out = length.out)

  # Modelo cosinor
  y <- M + A * cos(t + phi)

  # Opcionalmente graficar
  if (plot) {
    plot(t, y, type = "l", lwd = 2,
         xlab = "Fase (radianes)", ylab = "Expresion",
         main = "Curva Cosinor")
  }

  # Salida
  return(list(x = t, y = y))
}

# Funcion que detecta si la suma de un vector es igual a su longitud
and_logical <- function(data) {
  if (sum(data) == length(data))
    return(TRUE)
  else
    return(FALSE)
}

# Calcula la distancia angular m nima entre dos angulos
circ_dist <- function(a, b) {
  diff <- abs(a - b) %% (2 * pi)
  pmin(diff, 2 * pi - diff)
}
```

Listing B.1: Funciones auxiliares para filtrado y análisis circadiano

B.2. Preprocesamiento de los datos - CIRCUST

```r
# Anibal Hernando Novo

setwd("C:/ANIBALE/UNIVERSIDAD/TFG ESTADISTICA")

# Cargar librer as
library(cosinor)
library(ggplot2)
library(dplyr)
library(gridExtra)
library(readxl)
library(caret)
library(randomForest)
library(Matrix)
library(circular)
library(xgboost)
```

```r
library(FactoMineR)

# Cargar datos
rawDat <- read.table("C:/.../dat3.txt", header = TRUE, sep = "\t", row.
    names = 1)
nombres <- colnames(rawDat)
tipos <- sub("Exon_(\\d+)_.*", "\\1", nombres)
tabla_tipos <- table(tipos)
df_tipos <- as.data.frame(tabla_tipos)
colnames(df_tipos) <- c("Tipo de muestra (Exon)", "Numero de muestras")
print(df_tipos)

# Eliminar muestras del individuo Exon_7
rawDat <- rawDat[, -c(25,26,27)]

# Obtener nombres
geneName <- rownames(rawDat)
sampleName <- colnames(rawDat)

# Filtrar genes con muchos ceros
genes_ceros <- obtener_genes_con_muchos_ceros(rawDat)

# Calcular mCut para cada gen
cut <- floor(0.9 * ncol(rawDat))
mCut_values <- apply(rawDat, 1, function(row) mean(sort(as.numeric(row)
    )[1:cut]))

# Estadisticos y visualizacion
mCut_media <- mean(mCut_values)
mCut_mediana <- median(mCut_values)

hist(mCut_values, breaks = 100, main = "Distribucion de mCut",
     xlab = "mCut", col = "skyblue", border = "white")
abline(v = mCut_media, col = "red", lwd = 2, lty = 2)
abline(v = mCut_mediana, col = "darkgreen", lwd = 2, lty = 2)
text(x = mCut_media, y = par("usr")[4] * 0.9,
     labels = paste0("Media = ", round(mCut_media, 2)), col = "red",
         pos = 4)
text(x = mCut_mediana, y = par("usr")[4] * 0.85,
     labels = paste0("Mediana = ", round(mCut_mediana, 2)), col = "
         darkgreen", pos = 4)

# Leer genes core
library(readxl)
genes_core_54 <- read_excel("C:/.../54_genes_core.xlsx")$Gene.Symbol
genes_core <- c("PER1","PER2","PER3","CRY1","CRY2","CLOCK",
                "NR1D1","RORA","STAT3","TEF","ARNTL","DBP")

# Evaluar thresholds
seque <- seq(0, 5, by = 0.1)
res <- sapply(seque, test_threshold, mCut_values, rownames(rawDat),
    genes_core_54)
res12 <- sapply(seque, test_threshold, mCut_values, rownames(rawDat),
    genes_core)

umbral <- 0.4
idx_umbral <- which.min(abs(seque - umbral))
prop1 <- res[1, idx_umbral]
```

```
prop2 <- res[2, idx_umbral]
prop3 <- res12[2, idx_umbral]

par(mfrow = c(1,3))
plot(seque, res[1,], type = 'l', col = 'blue', lwd = 2,
     main = "Genes seleccionados", cex.main = 2,
     ylab = "Proporcion", xlab = "Umbral media recortada")
abline(v = umbral, col = 'red', lty = 2)
abline(h = prop1, col = 'red', lty = 2)
points(umbral, prop1, pch = 16, col = 'red')
text(umbral + 0.03, prop1 + 0.03, round(prop1, 3), col = "red", cex =
   2)

plot(seque, res[2,], type = 'l', col = 'darkgreen', lwd = 2,
     main = "54 genes core", cex.main = 2,
     ylab = "Proporcion", xlab = "Umbral media recortada")
abline(v = umbral, h = prop2, col = 'red', lty = 2)
points(umbral, prop2, pch = 16, col = 'red')
text(umbral + 0.05, prop2 - 0.02, round(prop2, 3), col = "red", cex =
   2)

plot(seque, res12[2,], type = 'l', col = 'orange', lwd = 2,
     main = "12 genes core", cex.main = 2,
     ylab = "Proporcion", xlab = "Umbral media recortada")
abline(v = umbral, h = prop3, col = 'red', lty = 2)
points(umbral, prop3, pch = 16, col = 'red')
text(umbral + 0.05, prop3 - 0.02, round(prop3, 3), col = "red", cex =
   2)

#Se comprueban resultados con el umbral en 0.4
table(abs(mCut_values) > 0.4)

#Se guardan los resultados finales para su posterior uso
resultado <- filtrarGenesPorMCut(rawDat, porcentaje = 0.9, threshold =
   0.4)
datos_filtrados <- resultado$data_filtrada
```

Listing B.2: Script de preprocesado de los datos

B.3. Ordenación y sincronización - CIRCUST

```
##########################################
# Step 1: Seleccion de genes core y normalizacion de la expresion
    genica
##########################################

genes_unordered_core<-as.matrix(datos_filtrados[genes_core,])
genes_unordered_core<-t(apply(genes_unordered_core, 1, normalize))

genes_unordered<-as.matrix(datos_filtrados)
genes_unordered<-t(apply(genes_unordered, 1, normalize))


##########################################################
```

```
# Step 2: Orden preliminar utilizando CPCA sobre la matriz de genes core
############################################################

p1<-PCA(t(genes_unordered_core),scale.unit = FALSE,graph=FALSE)
if(p1$eig[2,3]<25) warning("Los 2 primeros eigengenes explican poca variabilidad.")
pred<-predict(p1,newdata=t(genes_unordered_core))$coord[,1:2]
E1<-pred[,1]
E2<-pred[,2]
order1<-CPCA_order(E1,E2) %% (2*pi)

E1.order<-E1[order(order1)]
E2.order<-E2[order(order1)]

par(mfrow=c(2,2))
plot(sort(order1),E1.order, main="Eigengene 1",xlab="",ylab="")
plot(sort(order1),E2.order, main="Eigengene 2",xlab="",ylab="")
plot(E1.order,E2.order, main="1st vs 2nd Eigengene",xlab="Expresion E1",ylab="Expresion E2")
plot(cos(order1),sin(order1), main="Proyeccion de las muestras en el circulo unidad",xlab="cos",ylab="sin")

out <- funcionCosinor(E1.order,sort(order1))
pred <- out[[2]] + out[[3]] * cos(seq(0,2*pi,length.out=50) + out[[5]])

out2 <- funcionCosinor(E2.order,sort(order1))
pred2 <- out2[[2]] + out2[[3]] * cos(seq(0,2*pi,length.out=50) + out2[[5]])

par(mfrow=c(1,1))
plot(pred,pred2, main = "Los 2 primeros eigengenes realizan una trayectoria circular")

genes_ordered1<-genes_unordered[,order(order1)]
genes_ordered_core1<-genes_unordered_core[,order(order1)]
angle_order1<-sort(order1)

par(mfrow=c(3,4))
for( i in genes_core){
  plot(angle_order1,genes_ordered_core1[i,],
       xlab = "",ylab = "Expresion",xaxt='n',yaxt='n',xlim=c(0,2*pi))
  title(main=i)
  axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
       labels = c(expression(0), expression(pi/2), expression(pi),
          expression(3*pi/2), expression(2*pi)))
}

############################################################
# Step 3: Eliminacion de outliers
############################################################

threshold <- 2
out1<-which(sqrt(E1.order^2+E2.order^2)>threshold)
out1Names<-names(out1)
no_outliers1 <- !(colnames(genes_ordered1) %in% out1Names)
out1 <- which(!no_outliers1)

```

```r
plot_cpca_outliers <- function(E1, E2, sample_names = NULL, threshold) {
  distances <- sqrt(E1^2 + E2^2)
  outliers <- which(distances > threshold)
  par(mfrow = c(1, 1))
  plot(E1, E2, pch = 21, bg = "skyblue", col = "black",
       xlab = "CPCA E1", ylab = "CPCA E2",
       main = paste("Outliers obtenidos mediante CPCA (Threshold =",
          threshold, ")"))
  abline(h = 0, v = 0, col = "gray", lty = 2)
  theta <- seq(0, 2 * pi, length.out = 200)
  lines(threshold * cos(theta), threshold * sin(theta), col = "black")
  points(E1[outliers], E2[outliers], pch = 21, bg = "red", col = "black")
  if (!is.null(sample_names)) {
    short_names <- sub(".*_(S[0-9]+)_.*", "\\1", sample_names)
    text(E1[outliers], E2[outliers], labels = short_names[outliers],
         pos = 3, cex = 1, col = "darkred")
  }
  legend("topright", legend = c("Normal", "Outlier", "Umbral"),
         pch = c(1, 16, NA), lty = c(NA, NA, 1),
         col = c("black", "darkred", "black"), pt.cex = 0.8, cex = 0.8)
  return(outliers)
}
outliers <- plot_cpca_outliers(E1, E2, sample_names = sampleName,
   threshold = 2)

n_muestras <- ncol(genes_ordered_core1)
tiempos <- seq(0, 2*pi, length.out = n_muestras)
no_outliers <- apply(apply(genes_ordered_core1, 1, CosinorOutliers,
   time = tiempos),1,and_logical)
out2 <- which(!no_outliers)
no_outliers <- no_outliers[order(order(order1))]

############################################################
# Step 4: Eliminar outliers y volver a ejecutar CPCA
############################################################

genes_unordered<-genes_unordered[,no_outliers]
genes_unordered_core<-genes_unordered_core[,no_outliers]

genes_unordered<-t(apply(genes_unordered, 1, normalize))
genes_unordered_core<-t(apply(genes_unordered_core, 1, normalize))

p1<-PCA(t(genes_unordered_core),scale.unit = FALSE,graph=FALSE)
pred<-predict(p1,newdata=t(genes_unordered_core))$coord[,1:2]
E1<-pred[,1]
E2<-pred[,2]
order2<-CPCA_order(E1,E2) %% (2*pi)

genes_ordered2<-genes_unordered[,order(order2)]
genes_ordered_core2<-genes_unordered_core[,order(order2)]
angle_order2<-sort(order2)

E1.order2<-E1[order(order2)]
E2.order2<-E2[order(order2)]
par(mfrow=c(2,2))
plot(sort(order2),E1.order2, main="Eigengene 1",xlab="",ylab="")
```

```
plot(sort(order2),E2.order2, main="Eigengene 2",xlab="",ylab="")
plot(E1.order2,E2.order2, main="1st vs 2nd Eigengene",xlab="Expresion
   E1",ylab="Expresion E2")
plot(cos(order2),sin(order2), main="Proyeccion en el circulo unidad",
   xlab="",ylab="")

#Dibujar expresi n del gen con su R2
par(mfrow=c(3,4))
for( i in genes_core){
  plot(angle_order2,genes_ordered_core2[i,],
       xlab = "",ylab = "Expresion",xaxt='n',yaxt='n',xlim=c(0,2*pi))
  title(main=i)
  axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
       labels = c(expression(0), expression(pi/2), expression(pi),
          expression(3*pi/2), expression(2*pi)))
}

r2_table <- data.frame(Gen = character(), R2 = numeric(),
   stringsAsFactors = FALSE)
par(mfrow = c(3, 4), oma = c(0, 0, 4, 0))
for (gen in genes_core) {
  if (gen %in% rownames(genes_ordered2)) {
    x <- angle_order2
    y <- genes_ordered_core2[gen, ]
    fit_sync <- funcionCosinor(y, x)
    y_fit_sync <- fit_sync[[1]]
    phiEst_sync <- fit_sync[[5]]
    mest_sync <- fit_sync[[2]]
    aest_sync <- fit_sync[[3]]
    ss_res <- sum((y - y_fit_sync)^2)
    ss_tot <- sum((y - mean(y))^2)
    r2 <- 1 - ss_res / ss_tot
    r2_table <- rbind(r2_table, data.frame(Gen = gen, R2 = r2))
    plot(x, y, xlim = c(0, 2*pi), main = gen,
         xlab = "", ylab = "", xaxt = "n", yaxt = "n")
    axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
         labels = c(expression(0), expression(pi/2), expression(pi),
            expression(3*pi/2), expression(2*pi)))
    abline(v = (2*pi - phiEst_sync) %% (2*pi), col = "red", lty = 2)
    lines(seq(0, 2*pi, length.out = 200),
          generateCosinor(M = mest_sync, A = aest_sync, phi = phiEst_
             sync, plot = FALSE, length.out = 200)$y,
          col = "green", lwd = 2)
  }
}
mtext("Expresion de genes core (NO sincronizados)", outer = TRUE, cex =
    1.5, font = 2)
print(r2_table)
```

Listing B.3: Ordenacion y eliminacion de outliers en genes core

```
##############################################
# Step 5: Shynchronization
##############################################

fit_arntl <- funcionCosinor(genes_ordered_core2["ARNTL", ], angle_
   order2)
phiEst_arntl <- fit_arntl[[5]]
```

```
mest_arntl <- fit_arntl[[2]]
aest_arntl <- fit_arntl[[3]]
pico_arntl <- (2*pi-phiEst_arntl) %% (2*pi)

#Grafico de ARNTL sin sincronizar
par(mfrow=c(1,2))
plot(angle_order2,genes_ordered_core2["ARNTL",],type = "p", main = "ARNTL",
     xlab = "Fase circadiana (radianes)", ylab = "Expresion", xaxt = "n")
lines(seq(0, 2*pi, length.out = 200),
  generateCosinor(M = mest_arntl, A = aest_arntl, phi = phiEst_arntl, plot = FALSE, length.out = 200)$y,
  col = "green", lwd = 2)
abline(v = pico_arntl, col = "red", lty = 2, lwd = 2)
axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
     labels = c(expression(0), expression(pi/2), expression(pi), expression(3*pi/2), expression(2*pi)))

#Calculo del desfase de sincronizacion y de las nuevas fases
desfase <- (pi - pico_arntl)
angle_sync <- (angle_order2 + desfase) %% (2*pi)

fit_sync_arntl <- funcionCosinor(genes_ordered_core2["ARNTL", ],angle_sync)
phiEst_sync_arntl <- fit_sync_arntl[[5]]
mest_sync_arntl <- fit_sync_arntl[[2]]
aest_sync_arntl <- fit_sync_arntl[[3]]
pico_sync_arntl <- (2*pi-phiEst_sync_arntl) %% (2*pi)

#Grafico de ARNTL sincronizado
plot(angle_sync, genes_ordered_core2["ARNTL", ],
     type = "p", main = "ARNTL (sincronizado en pi)",
     xlab = "Fase circadiana sincronizada (radianes)", ylab = "Expresion",
     xaxt = "n")
lines(seq(0, 2*pi, length.out = 200),
      generateCosinor(M = mest_sync_arntl, A = aest_sync_arntl, phi = pico_sync_arntl, plot = FALSE, length.out = 200)$y,
      col = "green", lwd = 2)
abline(v = pico_arntl, col = adjustcolor("red", alpha.f = 0.3), lty = 2, lwd = 2)
abline(v = pico_sync_arntl, col = "red", lty = 2, lwd = 2)
axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
     labels = c(expression(0), expression(pi/2), expression(pi), expression(3*pi/2), expression(2*pi)))


# Graficos genes core sincronizados
par(mfrow = c(3, 4), oma = c(0, 0, 4, 0))
for (gen in genes_core) {
  if (gen %in% rownames(genes_ordered2)) {
    x <- angle_sync
    y <- genes_ordered_core2[gen, ]
    fit_sync <- funcionCosinor(y, x)
    y_fit_sync <- fit_sync[[1]]
    phiEst_sync <- fit_sync[[5]]
    mest_sync <- fit_sync[[2]]
```

```
    aest_sync <- fit_sync[[3]]
    plot(x, y, xlim = c(0, 2*pi), main = gen,
         xlab = "", ylab = "", xaxt = "n", yaxt = "n")
    axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
         labels = c(expression(0), expression(pi/2), expression(pi),
           expression(3*pi/2), expression(2*pi)))
    lines(seq(0, 2*pi, length.out = 200),
          generateCosinor(M = mest_sync, A = aest_sync, phi = phiEst_
            sync, plot = FALSE, length.out = 200)$y,
          col = "green", lwd = 2)
  }
}
mtext("Expresion de genes core (sincronizados)",
      outer = TRUE, cex = 1.5, font = 2)

#Guardo los resultados
saveRDS(angle_sync,"C:/ANIBALE/UNIVERSIDAD/CURSO 5  /TFG ESTADISTICA/
   Datos/angle_sync.rds")
saveRDS(genes_ordered2,
        file= "C:/ANIBALE/UNIVERSIDAD/CURSO 5  /TFG ESTADISTICA/Datos/
           genes_ordered2.rds")
saveRDS(genes_ordered_core2,
        file= "C:/ANIBALE/UNIVERSIDAD/CURSO 5  /TFG ESTADISTICA/Datos/
           genes_ordered_core2.rds")
```

Listing B.4: Sincronizacion de la expresión génica mediante ajuste Cosinor

B.4. Preprocesamiento y ordenación - CYCLOPS

```
# Leer archivos
cyclops_ordered <- read.csv("C:/ANIBALE/UNIVERSIDAD/TFG ESTADISTICA/
   Datos/CYCLOPS_ordered_expression.csv", row.names = 1)
order_cyc <- scan("C:/ANIBALE/UNIVERSIDAD/TFG ESTADISTICA/Datos/
   CYCLOPSmuscleExonCore12Phi.txt")

# Diagrama de fases circadianas estimadas
plot(as.circular(order_cyc), main= 'Fases circadianas estimadas con
   CYCLOPS')
plot(as.circular(angle_cir_sync), main= 'CIRCUST CON 12 GENES CORE')

# Comprobar que tengan las mismas muestras
dim(cyclops_ordered)
dim(circust_ordered)

# Normalizacion min-max [-1,1] a los datos de CYCLOPS
cyclops_ordered <- t(apply(cyclops_ordered, 1, normalize))

# Inicializar data frame para guardar los resultados
r2_cyc_table <- data.frame(Gen = character(), R2 = numeric(),
   stringsAsFactors = FALSE)

# Graficos genes core no sincronizados de CYCLOPS
coreG <- c("PER1","PER2","PER3","CRY1","CRY2","CLOCK","NR1D1","RORA","
   STAT3","TEF","ARNTL","DBP")
par(mfrow = c(3, 4), oma = c(0, 0, 4, 0))
```

```r

for (gen in coreG) {
  if (gen %in% rownames(cyclops_ordered)) {

    # Datos: expresion y fases sincronizadas
    x <- sort(order_cyc)
    y <- as.numeric(cyclops_ordered[gen, ])

    # Cosinor
    fit_sync <- funcionCosinor(y, x)
    y_fit_sync <- fit_sync[[1]]
    phiEst_sync <- fit_sync[[5]]
    mest_sync <- fit_sync[[2]]
    aest_sync <- fit_sync[[3]]

    # Calcular R
    ss_res <- sum((y - y_fit_sync)^2)
    ss_tot <- sum((y - mean(y))^2)
    r2 <- 1 - ss_res / ss_tot

    # Guardar R
    r2_cyc_table <- rbind(r2_cyc_table, data.frame(Gen = gen, R2 = r2))

    # Graficar
    plot(x, y, xlim = c(0, 2*pi), main = gen, cex.main = 1.5,
         xlab = "", ylab = "", xaxt = "n", yaxt = "n")

    axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
         labels = c(expression(0), expression(pi/2), expression(pi),
            expression(3*pi/2), expression(2*pi)))

    lines(seq(0, 2*pi, length.out = 200),
          generateCosinor(M = mest_sync, A = aest_sync, phi = phiEst_
              sync, plot = FALSE, length.out = 200)$y,
          col = "green", lwd = 2)
  }
}

# Titulo general al conjunto de graficos
mtext("Expresion de genes core",
      outer = TRUE, cex = 1.5, font = 2)

# Ver la tabla de resultados de R2
print(r2_cyc_table)
```

Listing B.5: Preprocesado y ordenacion de los datos de CYCLOPS

B.5. Comparación de CIRCUST vs CYCLOPS

```r


# COMPARACION DE ORDENDES GRAFICOS VALORES REALES VS ESTIMADOS

par(mfrow = c(1, 2))
require(CHIRAL)
```

```
7
8  # CIRCUST
9  nombres <- names(angle_cir_sync)
10 horas_char <- sub(".*_", "", nombres)
11 horas <- as.numeric(horas_char)
12 horas_rad <- (horas / 24) * 2 * pi
13 valores <- as.numeric(angle_cir_sync)
14
15 inf_phi = delta.phi(horas_rad, valores, mode = "say", median_scale = 12
       /pi)
16 abs(cor.c(horas_rad, inf_phi))
17 adj_phi = adjust.phases(horas_rad, inf_phi)
18 plot(horas_rad, adj_phi, col = "black",
19      xlab = "Hora real (radianes)", ylab = "Fase estimada (radianes)",
20      main = "CIRCUST: hora real vs. fase estimada", xaxt = "n")
21 abline(0, 1, col = "red", lty = 2)
22 axis(1, at = c(0:6))
23
24 mae_circular <- mean(circ_dist(horas_rad, adj_phi))
25 mae_horas <- mae_circular * (24 / (2 * pi))
26
27 cat("MAE circular:", round(mae_circular, 3), "radianes\n")
28 cat("MAE en horas:", round(mae_horas, 2), "h\n")
29
30
31
32 # CYCLOPS
33 nombres <- colnames(cyclops_ordered)
34 horas_char <- sub(".*_", "", nombres)
35 horas <- as.numeric(horas_char)
36 horas_rad <- (horas / 24) * 2 * pi
37 valores <- as.numeric(order_cyc)
38
39 inf_phi = delta.phi(horas_rad, valores, mode = "say", median_scale = 12
       /pi)
40 abs(cor.c(horas_rad, inf_phi))
41 adj_phi = adjust.phases(horas_rad, inf_phi)
42 plot(horas_rad, adj_phi, col = "black",
43      xlab = "Hora real (radianes)", ylab = "Fase estimada (radianes)",
44      main = "CYCLOPS: hora real vs. fase estimada",
45      xlim = c(0, 5.5), ylim = c(-3, 6),
46      xaxt = "n", yaxt = "n")
47 abline(0, 1, col = "red", lty = 2)
48 axis(1, at = c(0:6))
49 axis(2, at = -3:6)
50
51 mae_cyc <- mean(circ_dist(horas_rad, valores))
52 mae_horas <- mae_cyc * (24 / (2 * pi))
53
54 cat("MAE circular:", round(mae_cyc, 3), "radianes\n")
55 cat("MAE en horas:", round(mae_horas, 2), "h\n")
56
57
58 #COMPARACION GRAFICA DE GENES CORE
59
60 for (gen in coreG) {
61   if (gen %in% rownames(cyclops_ordered) && gen %in% rownames(circust_
       ordered)) {
```

```
62
63      par(mfrow = c(1, 2), oma = c(0, 0, 3, 0), mar = c(4, 4, 2, 1))
64
65      # CYCLOPS
66      x_cyc <- angle_cyc_sync
67      y_cyc <- as.numeric(cyclops_ordered[gen, ])
68      fit_cyc <- funcionCosinor(y_cyc, x_cyc)
69      yfit_cyc <- fit_cyc[[1]]
70      phi_cyc <- fit_cyc[[5]]
71      mest_cyc <- fit_cyc[[2]]
72      aest_cyc <- fit_cyc[[3]]
73      r2_cyc <- 1 - sum((y_cyc - yfit_cyc)^2) / sum((y_cyc - mean(y_cyc))
            ^2)
74
75      plot(x_cyc, y_cyc, xlim = c(0, 2*pi), main = paste0("CYCLOPS - ",
            gen, "\nR2 = ", round(r2_cyc, 3)),
76           pch = 16, xaxt = "n", yaxt = "n", xlab = "", ylab = "")
77      axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
78           labels = c(expression(0), expression(pi/2), expression(pi),
                expression(3*pi/2), expression(2*pi)))
79      abline(v = (2*pi - phi_cyc) %% (2*pi), col = "red", lty = 2)
80      lines(seq(0, 2*pi, length.out = 200),
81            generateCosinor(mest_cyc, aest_cyc, phi_cyc, plot = FALSE,
                  length.out = 200)$y,
82            col = "green", lwd = 2)
83
84      # CIRCUST
85      x_cir <- angle_cir_sync
86      y_cir <- as.numeric(genes_ordered2[gen, ])
87      fit_cir <- funcionCosinor(y_cir, x_cir)
88      yfit_cir <- fit_cir[[1]]
89      phi_cir <- fit_cir[[5]]
90      mest_cir <- fit_cir[[2]]
91      aest_cir <- fit_cir[[3]]
92      r2_cir <- 1 - sum((y_cir - yfit_cir)^2) / sum((y_cir - mean(y_cir))
            ^2)
93
94      plot(x_cir, y_cir, xlim = c(0, 2*pi), main = paste0("CIRCUST - ",
            gen, "\nR  = ", round(r2_cir, 3)),
95           pch = 16, xaxt = "n", yaxt = "n", xlab = "", ylab = "")
96      axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
97           labels = c(expression(0), expression(pi/2), expression(pi),
                expression(3*pi/2), expression(2*pi)))
98      abline(v = (2*pi - phi_cir) %% (2*pi), col = "red", lty = 2)
99      lines(seq(0, 2*pi, length.out = 200),
100           generateCosinor(mest_cir, aest_cir, phi_cir, plot = FALSE,
                  length.out = 200)$y,
101           col = "green", lwd = 2)
102
103     mtext(paste("Comparacion sincronizada para el gen:", gen), outer =
            TRUE, cex = 1.4, font = 2)
104     readline(prompt = "Presiona [Enter] para continuar al siguiente gen
            ...")
105   }
106 }
```

Listing B.6: Comparacion de ordenes circadianos y ajuste Cosinor en genes core

B.6. Identificación de ritmicidad - CIRCUST

```

### FUNCION DE RITMICIDAD ###

analizar_genes_cosinor_radianes <- function(data, fases_radianes) {
  if (length(fases_radianes) != ncol(data)) stop("Las fases deben tener la misma longitud que el numero de columnas (muestras)")
  expr_matrix <- data
  resultados <- data.frame(
    Gen = rownames(data),
    Mesor = NA,
    Amplitud = NA,
    Acrofase_radianes = NA,
    Pvalor = NA,
    Pvalor_ajustado = NA,
    Ritmico = NA
  )

  for (i in seq_len(nrow(expr_matrix))) {
    expr <- as.numeric(expr_matrix[i, ])
    if (any(is.na(expr)) || sd(expr) == 0) next
    df <- data.frame(expression = expr, time_var = fases_radianes)

    tryCatch({
      modelo <- cosinor.lm(expression ~ time(time_var), data = df, period = 2*pi)
      resumen <- summary(modelo)
      mesor <- resumen$transformed.table$estimate[1]
      amp <- resumen$transformed.table$estimate[2]
      acr_rad <- resumen$transformed.table$estimate[3]
      pval <- resumen$transformed.table$p.value[2]

      resultados$Mesor[i] <- round(mesor, 4)
      resultados$Amplitud[i] <- round(amp, 4)
      resultados$Acrofase_radianes[i] <- round(acr_rad, 3)
      resultados$Pvalor[i] <- pval
      resultados$Ritmico[i] <- !is.na(pval) && pval < 0.05

    }, error = function(e) {
      resultados$Pvalor[i] <- NA
    })
  }

  resultados$Pvalor_ajustado <- p.adjust(resultados$Pvalor, method = "BH")
  resultados$Ritmico <- ifelse(!is.na(resultados$Pvalor_ajustado) & resultados$Pvalor_ajustado < 0.1, TRUE, FALSE)
  return(resultados)
}

# Ejecutar el analisis
res_circust <- analizar_genes_cosinor_radianes(circust_ordered, angle_cir_sync)
table(res_circust$Ritmico)

# Genes ritmicos detectados
```

```r
rit_SI_circust <- res_circust %>% filter(Ritmico == TRUE) %>% select(Gen)
count(rit_SI_circust)



# 1. % de genes ritmicos en 54 genes
genes_core_54 <- "C:/ANIBALE/UNIVERSIDAD/TFG ESTADISTICA/Datos/54_genes_core.xlsx"
genes_ref <- read_excel(genes_core_54, sheet = 1)$Gene.Symbol
genes_ref_filtrados <- intersect(genes_ref, rownames(circust_ordered))
genes_cir_detect <- rit_SI_circust$Gen
genes_cir_comun <- intersect(genes_ref_filtrados, genes_cir_detect)
porcentaje_cir <- length(genes_cir_comun) / length(genes_ref_filtrados) * 100

cat(length(genes_cir_comun), "de los", length(genes_ref_filtrados),
    "genes de referencia estan presentes en Circust (",
    round(porcentaje_cir, 2), "%)\n")
cat("\n Genes detectados:\n")
print(sort(genes_cir_comun))


# 2. R2 de genes ritmicos
genes_detectados <- intersect(genes_ref_filtrados, genes_cir_comun)
r2_detectados <- sapply(genes_detectados, function(gen) {
  if (!(gen %in% rownames(circust_ordered))) return(NA)
  y <- as.numeric(circust_ordered[gen, ])
  fit <- funcionCosinor(y, angle_sync)
  yfit <- fit[[1]]
  ss_res <- sum((y - yfit)^2)
  ss_tot <- sum((y - mean(y))^2)
  r2 <- 1 - ss_res / ss_tot
  return(r2)
})

mean_r2 <- mean(r2_detectados, na.rm = TRUE)
cat("R2 promedio de los 37 genes detectados:", round(mean_r2, 3), "\n")
```

Listing B.7: Identificacion de ritmicidad mediante modelo Cosinor

B.7. Análisis de ritmicidad con algortimos de ML

Se utiliza validación cruzada implementada mediante el parámetro `trainControl(method = ¢v", number = 5)` de la función `train()` del paquete `caret`.

```r

### ALGORITMOS DE ML ###

# Preparar el dataset
# Convertir la matriz ordenada a data frame y a adir la columna de nombres de genes
circust_ordered <- as.data.frame(circust_ordered)
circust_ordered$Gen <- rownames(circust_ordered)

```

```r
# Renombrar columna "Symbol" a "Gen" en la tabla de resultados de ritmicidad
colnames(res_circust)[colnames(res_circust) == "Symbol"] <- "Gen"

# Seleccionar columnas relevantes: nombre del gen y valor l gico de ritmicidad
ritmicidad <- res_circust[, c("Gen", "Ritmico")]

# Unir los datos de expresi n con los resultados de ritmicidad
circust_merged <- merge(circust_ordered, ritmicidad, by = "Gen", all.x = TRUE)

# Mostrar primeras filas y dimensiones del nuevo conjunto
head(circust_merged)
dim(circust_merged)

# Preparar el dataset final para modelado
df_ml <- as.data.frame(circust_merged)
df_ml$Ritmico <- as.factor(df_ml$Ritmico)  # Convertir a variable categ rica

# Separar genes core (de referencia) y el resto
genes_core <- circust_merged %>% filter(Gen %in% genes_ref)
otros_genes <- circust_merged %>% filter(!(Gen %in% genes_ref))

# Fijar semilla para reproducibilidad y crear partici n aleatoria del 20% para test
set.seed(123)
testIndex_restantes <- createDataPartition(otros_genes$Ritmico, p = 0.2, list = FALSE)

# Conjunto de test: genes core + 20% aleatorio del resto
test <- rbind(genes_core, otros_genes[testIndex_restantes, ])

# Conjunto de entrenamiento: el 80% restante
train <- otros_genes[-testIndex_restantes, ]

# Comprobar si todos los genes core est n presentes en el conjunto de test
genes_core_en_test <- genes_ref[genes_ref %in% test$Gen]
cat("Genes core presentes en test:", length(genes_core_en_test), "de 45\n")

# Mostrar si falta alg n gen core
genes_faltantes <- setdiff(genes_ref, test$Gen)
if (length(genes_faltantes) > 0) {
  cat("Los siguientes genes core NO estan en test:\n")
  print(genes_faltantes)
} else {
  cat("Todos los genes core estan presentes en el conjunto de test.\n")
}


# Preparar datasets de modelado
df_alg_train <- train %>% select(where(is.numeric), Ritmico)
df_alg_test  <- test %>% select(where(is.numeric), Ritmico)

control <- trainControl(method = "cv", number = 5)
```

```
59
60 # Regresion logistica
61 modelo_log <- train(
62   Ritmico ~ ., data = df_alg_train,
63   method = "glm", family = "binomial",
64   trControl = control
65 )
66 pred_log <- predict(modelo_log, newdata = df_alg_test)
67 resultados_log <- data.frame(Gen = test$Gen, Real = test$Ritmico,
       Predicho = pred_log)
68 saveRDS(resultados_log, file = "resultados_log.rds")
69
70 # Random Forest
71 modelo_rf <- train(
72   Ritmico ~ ., data = df_alg_train,
73   method = "rf",
74   trControl = control,
75   importance = TRUE
76 )
77 pred_rf <- predict(modelo_rf, newdata = df_alg_test)
78 resultados_rf <- data.frame(Gen = test$Gen, Real = test$Ritmico,
       Predicho = pred_rf)
79 saveRDS(resultados_rf, file = "resultados_rf.rds")
80
81 # SVM radial
82 modelo_svm <- train(
83   Ritmico ~ ., data = df_alg_train,
84   method = "svmRadial",
85   trControl = control,
86   preProcess = c("center", "scale"),
87   tuneLength = 5
88 )
89 pred_svm <- predict(modelo_svm, newdata = df_alg_test)
90 resultados_svm <- data.frame(Gen = test$Gen, Real = test$Ritmico,
       Predicho = pred_svm)
91 saveRDS(resultados_svm, file = "resultados_svm.rds")
92
93 # XGBoost
94 modelo_xgb <- train(
95   Ritmico ~ ., data = df_alg_train,
96   method = "xgbTree",
97   trControl = control,
98   tuneLength = 5
99 )
100 pred_xgb <- predict(modelo_xgb, newdata = df_alg_test)
101 resultados_xgb <- data.frame(Gen = test$Gen, Real = test$Ritmico,
       Predicho = pred_xgb)
102 saveRDS(resultados_xgb, file = "resultados_xgb.rds")
103
104 # Resultados
105 cat("Regresion Logistica:\n")
106 print(confusionMatrix(pred_log, test$Ritmico))
107 conf_log <- confusionMatrix(pred_rf, test$Ritmico)
108 capture.output(conf_log, file = "confusion_log.txt")
109
110 cat("\nRandom Forest:\n")
111 print(confusionMatrix(pred_rf, test$Ritmico))
112 conf_rf <- confusionMatrix(pred_rf, test$Ritmico)
```

```
capture.output(conf_rf, file = "confusion_rf.txt")

cat("\nSVM Radial:\n")
print(confusionMatrix(pred_svm, test$Ritmico))
conf_svm <- confusionMatrix(pred_svm, test$Ritmico)
capture.output(conf_svm, file = "confusion_svm.txt")

cat("\nXGBoost:\n")
print(confusionMatrix(pred_xgb, test$Ritmico))
conf_xgb <- confusionMatrix(pred_xgb, test$Ritmico)
capture.output(conf_xgb, file = "confusion_xgb.txt")
```

Listing B.8: Algoritmos de aprendizaje supervisado aplicados a la prediccion de ritmicidad