Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

Metodología para el análisis de expresiones génicas utilizando modelos de señales oscilatorias. Aplicación a datos de músculo humano

Autor: Samuel Sjur Bassols Citores

Tutoras: Yolanda Larriba González, Itziar Fernández Martínez

Año: 2023

Agradecimientos

Quisiera dar las gracias a mis tutoras Yolanda Larriba e Itziar Fernández por proponerme un tema para el TFG tan interesante y relevante. También quiero agradecerles lo mucho que me han enseñado durante estos meses, su inestimable ayuda y su infinita paciencia.

Gracias al Consejo Social de la Universidad de Valladolid por otorgarme una beca del Programa de Colaboración que me ha permitido adentrarme en el fascinante mundo de la investigación.

También, quisiera dar las gracias a mi familia, por sus ánimos y apoyo incondicional durante toda mi vida.

Resumen

Los ritmos circadianos en los seres vivos están controlados por un sistema de sincronización interno regulado a nivel transcripcional que genera redes de genes que oscilan en ciclos de 24 horas. Estos ritmos juegan un papel importante en los procesos biológicos, por lo que el estudio de los patrones de expresión de genes rítmicos cobra gran interés en investigación biomédica. En los estudios genéticos humanos, la repetición consecutiva de biopsias conlleva un grave riesgo. Por ello, habitualmente se trabaja con datos de expresiones post-mortem de distintos donantes para un mismo tejido, donde el instante de tiempo en el que se toman las muestras es generalmente desconocido o impreciso. Por tanto, el análisis de ritmicidad de expresiones génicas requiere resolver el problema de estimación del orden temporal, previo al estudio de los patrones rítmicos. Este trabajo propone una metodología para el análisis de expresiones de genes post-mortem en tejido muscular en humanos de la base de datos GTEX basada en el modelo FMM para el análisis de señales oscilatorias. El marco propuesto permite abordar de forma íntegra el análisis de ritmicidad en datos de expresión génica, desde el preprocesado hasta ofrecer una posible explicación biológica de los patrones rítmicos asociados a los genes, pasando por el problema de la estimación del orden.

Palabras clave: modelos FMM, señales oscilatorias, datos circulares, expresión génica, ritmo circadiano.

Abstract

Circadian rhythms in living beings are controlled by an internal synchronization system regulated at the transcriptional level that generates gene networks that oscillate in 24-hour cycles. These rhythms play an important role in biological processes, so the study of rhythmic gene expression patterns is of great interest in biomedical research. In human genetic studies, the consecutive repetition of biopsies carries a serious risk. Therefore, we usually work with post-mortem expression data from different donors for the same tissue, where the timepoint at which the samples are taken is generally unknown or imprecise. Therefore, the analysis of gene expression rhythmicity requires solving the problem of estimating the temporal order prior to the study of the rhythmic patterns. This work proposes a methodology for the analysis of post-mortem gene expressions in human muscle tissue from the GTEX database based on the FMM model for the analysis of oscillatory signals. The proposed framework allows a comprehensive approach to the analysis of rhythmicity in gene expression data, from preprocessing to offering a possible biological explanation of the rhythmic patterns associated with genes, including the problem of order estimation.

Key words: FMM models, oscillatory signals, circular data, gene expression, circadian rhythm.

Índice general

Índice de figuras

Índice de tablas

Abreviaturas

- BP: Proceso Biológico (*Biological Process*)
- CC: Componente celular (*Cellular Component*)
- CPCA: Análisis de componentes principales circulares (*Circular Principal Component Analysis*)
- DNA: Ácido desoxirribonucleico (*Deoxyribonucleic acid*)
- ES: *Enrichment Score*
- FDR: Tasa de falsos descubrimientos (*False Discovery Rate*)
- FMM: *Frequency Modulated Möbius*
- FPCA: Análisis de componentes principales funcionales (*Functional Principal Component Analysis*)
- GO: *Gene Ontology*
- GSEA: *Gene Set Enrichment Analysis*
- GTEX: *Genotype-Tissue Expression*
- MESOR: *Midline Estimating Statistic Of Rhythm*
- MF: Función molecular (*Molecular Function*)
- PCA: Análisis de componentes principales (*Principal Component Analysis*)
- RNA: Ácido ribonucleico (*Ribonucleic acid*)
- SVD: Descomposición en valores singulares (*Singular value decomposition*)
- TOD: Hora de la muerte (*Time of death*)
- TSP: Problema del vendedor viajante (*Travelling Salesman Problem*)

Introducción

En la gran mayoría de seres vivos se observa ritmicidad en los procesos biológicos que desempeñan y que les rodean. La mayoría de estos procesos están controlados por lo que se conoce como ritmos circadianos. Estos ritmos regulan la actividad de los distintos procesos en función del ciclo luz-oscuridad, de tal manera que procesos como el hambre o el sueño se ajustan a este ciclo en función de las necesidades de la especie. Esta ritmicidad no está solo presente en procesos biológicos que son observables a gran escala, sino que también se observa en procesos como la temperatura corporal, la tensión arterial, el metabolismo o los niveles hormonales [Levi and Schibler, 2007, Curtis and FitzGerald, 2006]. De forma más precisa, los ritmos circadianos en los seres vivos están dirigidos por un sistema de sincronización interno regulado a nivel transcripcional que genera redes de genes que oscilan con patrones de expresión oscilatorios a lo largo de las 24 horas del día. La importancia de los estudios en este campo se puede ver ejemplificada en el Premio Nobel de 2017 en fisiología o medicina, otorgado a Jeffrey C. Hall, Michael Rosbash y Michael W. Young por sus descubrimientos de los mecanismos moleculares que controlan el ritmo circadiano [Press Release, 2017].

El análisis de la ritmicidad a nivel molecular se realiza a partir de los patrones de expresión de los genes. La expresión génica es el proceso por el que la información codificada en el DNA se decodifica para producir un producto génico. Este producto génico serán moléculas de RNA que a su vez, pueden codificar para proteínas o para otras moléculas de RNA que cumplen otras funciones. Por tanto, se puede entender como el mecanismo que controla cuándo y dónde se producen moléculas de RNA y proteínas, así como la cantidad de estos productos que se producen. El RNA y las proteínas producidas por muchos genes sirven para regular la expresión de otros genes y algunos de estos genes tienen un comportamiento de expresión cíclico, ya que se encargan de regular el ritmo circadiano. En la Figura 1.1 se muestra de manera muy simplificada la secuencia de acontecimientos durante una oscilación de 24 h del gen Periodo (*PER*), el primer gen descrito que controla el ritmo circadiano. La proteína codificada por el gen Periodo, se acumula durante la noche y se degrada durante el día. Así pues, los niveles de proteína PER oscilan a lo largo de un ciclo de 24 horas, en sincronía con el ritmo circadiano [Press Release, 2017].

La forma más directa para determinar el perfil de expresión génica de las células sería medir la cantidad de proteínas funcionales que se producen en un momento dado. Sin embargo, la gran variedad de localizaciones, modificaciones y contextos en los que se encuentran, así como la falta de información sobre el proteoma completo de un organismo, hace que sea un procedimiento no factible por el momento. En su lugar, se considera que los niveles de mRNA constituyen una buena aproximación y resultan

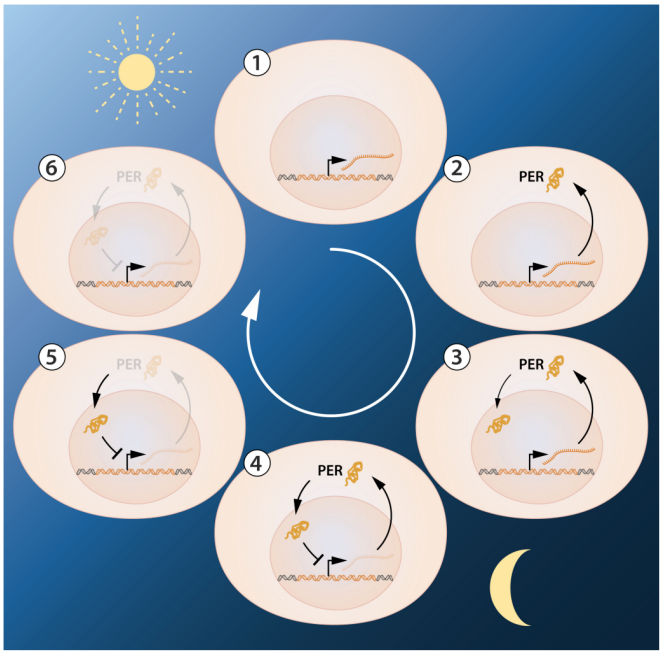


Figura 1.1: Ilustración simplificada del funcionamiento del gen Periodo durante una oscilación de 24 h.

(1) Cuando el gen Periodo está activo, se produce mRNA. (2) El mRNA se transporta al citoplasma de la célula donde se traduce a la proteína PER. (3) La proteína PER se acumula en el núcleo de la célula por la noche y (4) bloquea la actividad del gen Periodo que deja de expresar la proteína PER (5) y la que hay se degrada durante el día (6) con lo que puede transcribirse de nuevo el gen Periodo (1). Fuente [Press Release, 2017].

mucho más fáciles de medir. Estos niveles de mRNA son lo que se conoce como datos de expresión génica. Actualmente, existen básicamente dos tecnologías para producir esta información a gran escala: microarrays y RNA-seq. Ambas tecnologías tratan de determinar los niveles de expresión de miles de genes, pero, mientras que en el caso de los microarrays estos genes deben preseleccionarse, con RNA-seq se es capaz de secuenciar todo el genoma. La tecnología de RNA-seq ha facilitado la secuenciación masiva que permite disponer de bases de datos de alta dimensión con información de expresión de genes. Una de estas bases de datos es *Genotype-Tissue Expression* (GTEX) [Lonsdale et al., 2013] que recopila información de 17382 muestras post-mortem de 948 donantes para 54 tejidos humanos. Además de las expresiones para cada gen, GTEX contiene información demográfica y clínica de cada donante, como la edad, el sexo, la raza o la causa de muerte. Es de esperar que para cada tejido exista una colección de genes con un comportamiento rítmico.

El análisis de ritmicidad a nivel molecular está sujeto a diversas fuentes de variabilidad, tanto extrínsecas, errores en los aparatos de medida o en la tecnología empleada, como intrínsecas, debido a la variabilidad inter-individuo en la expresión de un mismo gen para un mismo tejido. Además, los estudios genéticos que involucran muestras post-mortem como GTEX, cuentan con una dificultad añadida que debe ser abordada antes del análisis de ritmicidad. En estos estudios es habitual que se desconozca, o bien se tengan estimaciones imprecisas, de la hora de la muerte (TOD, del inglés *Time of Death*) con las que reconstruir la expresión genética. Por lo que el primer problema a abordar en estos casos es el de la estimación del orden temporal, ver Figura 1.3.

Como se indica en [Larriba et al., 2022], el problema de estimación del orden temporal puede formularse matemáticamente como la búsqueda de un vector m-dimensional que proporcione lo que se conoce como orden circular (Figura 1.2). Una ordenación circular representa hasta $2m$ configuraciones distintas del tiempo de recogida de muestras a lo largo del día de 24 horas, dependiendo de la elección del punto de partida y de la orientación horaria o antihoraria. La elección de la direccionalidad es decisiva ya que desempeña un papel clave en la correcta identificación de la temporalidad de los procesos biológicos a lo largo del día. A este proceso se le conoce como sincronización.

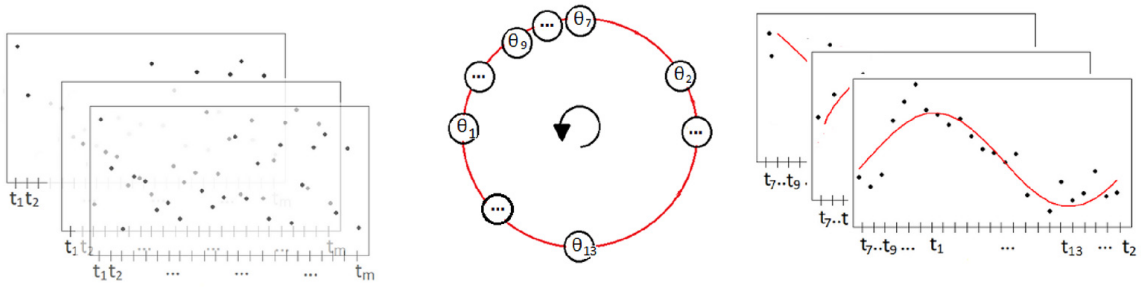


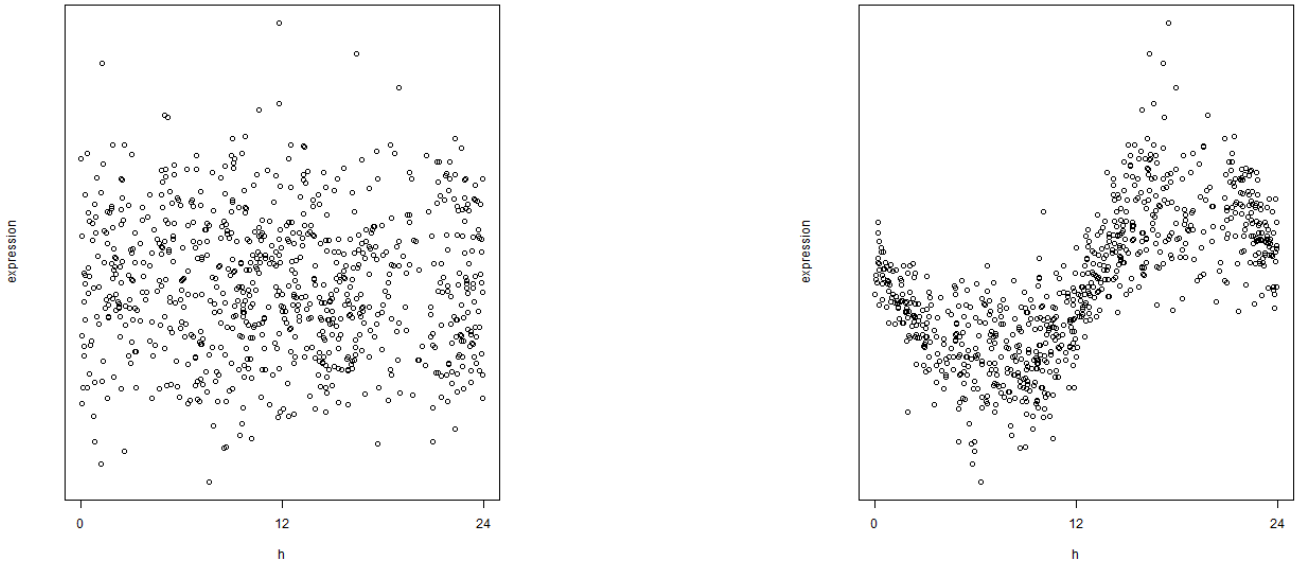
Figura 1.2: Esquema del problema de estimación del orden circular.

Izquierda: Datos de expresión génica desordenados a lo largo de un día. Centro: Orden circular en el que se considera la dirección antihoraria. Derecha: Expresión génica ordenada a lo largo del día de 24 horas. Fuente [Larriba et al., 2022].

El problema de estimación del orden temporal ha recibido bastante atención en los últimos años, ya que es un problema clave previo a la identificación de genes rítmicos. En los últimos años han aparecido metodologías como Oscope [Leng et al., 2015], reCAT [Liu et al., 2017] o CYCLOPS [Anafi et al., 2017]. Oscope y reCat son metodologías para recuperar las dinámicas del ciclo celular a partir de la información no sincronizada del transcriptoma de una única célula. Por otro lado, CYCLOPS necesita información sobre la expresión de genes homólogos de otras especies estudiadas para obtener el orden. También se ha propuesto una solución a partir de la formulación del problema como una minimización en el problema del vendedor viajante (TSP) en [Larriba et al., 2020], pero este tiene el inconveniente de que se obtiene un orden equiespaciado entre los distintos puntos temporales.

Se han propuesto otras metodología para abordar el problema del orden en la base de GTEX, como con modificaciones a CYCLOPS en [Ruben et al., 2018], pero solo se ha desarrollado sobre una selección limitada de tejidos de la base GTEX.

Una vez se ha estimado el orden temporal de las muestras, se puede llevar a cabo un análisis de ritmicidad a partir de los patrones de expresión génica. El objetivo principal es determinar qué genes, entre más de 20000, presentan patrones de expresión oscilatorios y son por tanto identificados como rítmicos. Identificar este tipo de genes es clave para comprender una gran variedad de procesos biológicos, puesto que el momento de máxima expresión de estos genes está relacionado con el momento del día en que el gen realiza su función biológica. Sin embargo, estos genes pueden adoptar una gran variedad de patrones y como es habitual en los estudios génicos suelen estar sujetos a una alta variabilidad.



(a) TOD

(b) Estimación del Orden

Figura 1.3: Comparación de la expresión génica ordenada a partir de TOD (a) y resolviendo el problema del orden temporal(b) para el gen *PER3*.

Una aproximación para estudiar la ritmicidad de los genes es tratar de ajustar modelos que permitan estimar el patrón cíclico. Algunos de los métodos utilizados para llevar a cabo estos ajustes son los siguientes:

- Cosinor [Cornelissen, 2014]: es un modelo paramétrico que ajusta una onda sinusoidal a la evolución de valores a lo largo del tiempo. Cosinor ha sido usado ampliamente en distintos campos de la fisiología. Sin embargo, este modelo podría ser demasiado rígido para estudiar expresiones en las que la onda no es simétrica [Rueda et al., 2019].

- FMM [Rueda et al., 2019]: Es un modelo parámetrico de 5 parámetros que permiten caracterizar la amplitud, localización y forma de la onda. Además su flexibilidad permite ajustar una amplia variedad de patrones oscilatorios. Este modelo es una generalización del modelo Cosinor, y por tanto ajusta mejor a patrones cuya onda no es simétrica.

- Modelos no paramétricos basados en regresión isotónica [Larriba et al., 2020], donde la modelización de la señal oscilatoria se realiza mediante restricciones de orden formuladas mediante desigualdades matemáticas.

También será interesante, una vez conocido el comportamiento de cada gen, poder agrupar genes con el mismo patrón cíclico. Disponer de una estimación de la onda para cada gen permite utilizar técnicas de análisis funcional y a partir de estas realizar un agrupamiento o *clustering* de los genes en función de sus patrones.

Una vez que se ha identificado un grupo de genes rítmicos es revelador conocer qué funciones biológicas desempeñan los genes del grupo, para poder así sacar conclusiones sobre los procesos biológicos que

regulan estos genes y las funciones que tienen en la fisiología celular. Para estudiar estas funcionalidades se usan bases de datos biológicas que recogen el conocimiento previo generado por la comunidad científica a lo largo de muy diferentes estudios. Una de las bases de datos más utilizada es *Gene Ontology* (GO) [Central et al., 2023, Ashburner et al., 2000], una ontología en que cada gen o producto génico se anota a una o varias categorías jerárquicamente relacionadas que recogen el conocimiento biológico sobre el mismo. Esta ontología es útil para identificar qué tienen en común un grupo de genes implicados en un determinado proceso. Los métodos de enriquecimiento funcional permiten identificar, a partir de un listado de genes de interés, qué términos de la ontología están sobre o infra representados en dicho listado. Uno de los métodos más populares es *Gene Set Enrichment Analysis* (GSEA) [Subramanian et al., 2005].

El objetivo de este trabajo es proponer una metodología general que permita resolver el problema de estimación temporal, identificar genes rítmicos y configurar agrupaciones de genes rítmicos según su función biológica. Esta metodología se ilustra para el tejido muscular de la base de expresiones de genes post-mortem GTEX.

Este trabajo consta de dos partes. Primero se proponen 3 métodos para obtener el orden de las muestras y se compara su funcionamiento mediante un estudio de simulación. El objetivo es seleccionar un método para obtener el orden temporal de las muestras que se pueda utilizar en la segunda parte, en la que se propone un algoritmo para ordenar las muestras, detectar los genes cíclicos, agruparles en función de la similitud que tienen sus patrones y estudiar las funciones y propiedades de estos grupos de genes.

1.1. Objetivos

El objetivo general de este trabajo es el desarrollo de un procedimiento para determinar la ritmicidad de genes en tejidos humanos y modelizar su comportamiento cíclico a partir de modelos de señales oscilatorias. Para este trabajo se proponen los siguientes objetivos específicos:

- Revisión del estado del arte: Métodos de estimación del orden temporal, modelos matemáticos para el análisis de señales oscilatorias, *clustering* de datos funcionales y enriquecimiento funcional.
- Planteamiento de una metodología completa para la obtención del orden temporal, detección de genes rítmicos a partir de modelos de señales oscilatorias y para la clasificación de genes rítmicos en función de su forma y estudio de sus características.
- Lectura, preprocesado y análisis de los datos de la base de datos GTEX.
- Desarrollo y validación de la metodología estadística e implementación de *software*.
- Explotación de los resultados.

1.2. Asignaturas relacionadas

- Computación Estadística: En esta asignatura se presentan los conocimientos necesarios acerca de la programación en el lenguaje R.

- Análisis de datos: Una parte importante de la asignatura se dedica a la comprensión del análisis de componentes principales.
- Regresión y Anova y Modelos lineales: En estas asignaturas se establecen los conocimientos para el ajuste de modelos de regresión y la comprensión de las distintas medidas de calidad de un modelo.
- Modelos de Investigación Operativa: En esta asignatura se explica la optimización del problema TSP, así como sus posibles usos para estimar el orden temporal de las muestras de expresiones génicas.
- Análisis Multivariante: En esta asignatura se explican distintos algoritmos de *clustering*.

1.3. Ampliación de los contenidos

La principal ampliación de contenidos viene dada por trabajar con nuevos tipos de datos que no se han planteado en el grado, los datos circulares y los datos funcionales necesarios para el análisis funcional. También el tipo de problema que se plantea resolver en este trabajo no es estándar y resulta complejo comparado con los que se han visto a lo largo del grado en los que para la mayoría se pueden validar los resultados. Los modelos de señales oscilatorias son otro aspecto que no se explora en el grado. Por último, también hay que destacar que el trabajo con bases de datos masivas y la correcta programación para optimizar los procesos, son una aportación adicional a los conocimientos.

Metodología

En esta sección se exponen las distintas metodologías que se utilizan en este trabajo. En la última sección se combina todo lo explicado en un algoritmo para la obtención de resultados.

2.1. Señal oscilatoria

Una señal oscilatoria es la medida de la trayectoria de una partícula en un sistema oscilatorio, en el que la partícula vuelve al estado inicial después de un periodo.

Matemáticamente una señal oscilatoria en el espacio euclídeo $\mu(t)$ se puede formular de manera equivalente como una señal $\phi(t)$ en el círculo unidad (espacio circular) de la siguiente forma [Larriba et al., 2020]:

$$\mu(t) = \cos(\phi(t)), 0 < \phi(t) \leq 2\pi, 0 < t \leq 2\pi \tag{2.1}$$

En la Figura 2.1 se muestra la formulación equivalente para una señal oscilatoria con la estructura *up-down-up* en el espacio euclídeo y que sigue un orden circular.

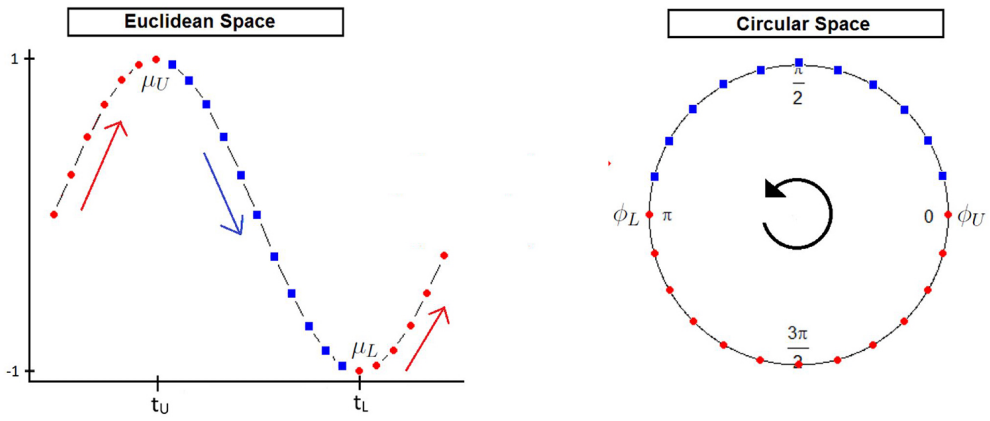


Figura 2.1: Esquematización de señal oscilatoria en el espacio circular y en el espacio euclídeo. Fuente [Larriba et al., 2020]

2.1.1. Datos circulares

Los datos circulares son aquellos que representan valores angulares. El análisis y cálculo de estadísticos para este tipo de datos no sigue los estándares de los datos lineales. Los datos circulares no se pueden representar solo mediante un soporte lineal de ángulos θ, ya que se está omitiendo información de la dirección y la distancia entre los datos. Así mismo, las operaciones aritméticas no están definidas de la misma forma para estos datos circulares. Y por ejemplo, para una transformación de la escala como las que se utilizan en este trabajo, se define la operación $\theta \bmod 2\pi$ para mantener el orden y obtener una representación lineal en el intervalo $[0, 2\pi]$.

Otro ejemplo más ilustrativo se expone en la Figura 2.2, donde el objetivo es encontrar la dirección media de dos valores circulares $\theta_1 = \frac{\pi}{12}$ y $\theta_2 = \frac{11\pi}{12}$. En la representación gráfica queda claro que la dirección media de ambos valores angulares es 0, mientras que la media aritmética de ambas componentes en una representación lineal resultaría en un valor medio de π, es decir el resultado en la dirección opuesta. Al realizar la media aritmética se ha perdido la información sobre la dirección.

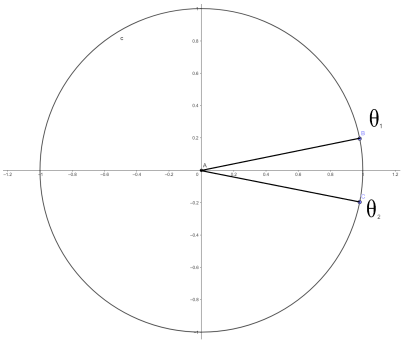


Figura 2.2: Ejemplo para ilustrar la diferencia entre la media aritmética y la circular.

A continuación vamos a definir los estadísticos media circular, media circular ponderada y desviación típica circular ponderada, de interés para este trabajo. Sean $\boldsymbol{\theta} = (\theta_1, ..., \theta_m)'$ un vector de m observaciones angulares, su media circular se define como:

$$\bar{\theta} = \arctan\left(\frac{1}{m}\sum_{j=1}^{m}\sin\theta_j, \frac{1}{m}\sum_{j=1}^{m}\cos\theta_j\right) \tag{2.2}$$

Sea $\boldsymbol{w} = (w_1, ..., w_m)'$ un vector de m pesos para cada observación de $\boldsymbol{\theta}$, se define la media circular con pesos $\boldsymbol{w} = (w_1, ..., w_m)$ como:

$$\bar{\theta}_w = \arctan\left(\frac{1}{\sum_{j=1}^{m} w_j}\sum_{j=1}^{m} w_j \sin\theta_j, \frac{1}{\sum_{j=1}^{m} w_j}\sum_{j=1}^{m} w_j \cos\theta_j\right) \tag{2.3}$$

La media circular es un caso particular de la media circular ponderada cuando todos los $w_j = 1$.

Por último, se define la desviación típica ponderada circular como:

$$\bar{s}_w(\theta) = \sqrt{-2\log(\sqrt{\frac{(\sum w_j \sin\theta_j)^2 + \sum w_j \cos\theta_j)^2}{\sum_{j=1}^{m} w_j}}} \tag{2.4}$$

Estos estadísticos y muchos otros para datos circulares se pueden encontrar en [Jammalamadaka and SenGupta, 2001].

2.2. Estimación del orden temporal

Sea $[X]$ una matriz de expresión de n genes a lo largo de m muestras, tal que un valor $x_{i,j}$ representa una observación del individuo j para el gen i, y Θ los posibles órdenes circulares de las muestras. El problema de estimación del orden temporal se formula matemáticamente como el siguiente problema que minimiza las distancias entre $\boldsymbol{\theta}$ y $[X]$:

$$\arg\min_{\boldsymbol{\theta}\in\Theta} d(\boldsymbol{\theta}, [X]) \tag{2.5}$$

Donde d denota la distancia considerada para el problema.

El problema de minimización es un problema NP-hard ya que existen $(m-1)!$ posibles órdenes.

En este proyecto se plantean 3 metodologías para aproximar este problema. Estas tres metodologías se comparan por simulación posteriormente en la sección 3.1 de los resultados.

2.2.1. Análisis de Componentes Principales Circulares (CPCA)

El método CPCA propone una aproximación simple y eficiente al problema de la estimación del orden. CPCA es un método de reducción de dimensionalidad no lineal que describe la estructura circular que subyace en los ritmos moleculares por medio de una proyección en el círculo unidad. La reducción de la dimensión de los genes produce componentes combinación lineal de genes en la dirección de máxima expresión que se denominan *eigengenes*.

Para la matriz de expresiones de genes no ordenada $[X]$, cada *eigengene* $\boldsymbol{E_i}, i = 1, ..., n$ se obtiene a partir de la descomposición en valores singulares SVD (del inglés *Singular Value Decomposition*) de la matriz $[X]$ [Alter et al., 2000]. Sea $[A] = [X \cdot X']$, según esta descomposición se tiene que $[A] = [U \cdot D \cdot D' \cdot V']$, donde $[U] = [V]$ ya que $[A]$ es una matriz simétrica y $[D \cdot D']$ es la matriz diagonal con los valores singulares de $[A]$. Por tanto, para obtener los *eigengenes* correspondientes a $[X]$ se tiene que proyectar $[U]$ y $[D]$ sobre la matriz de datos de la siguiente manera: $[U] = [X' \cdot V \cdot D^{-1}]$. En concreto, las columnas de la matriz $[U]$ se corresponden con los *eigengenes*, que son por tanto una combinación lineal que mejor representa la variabilidad presente en $[X]$.

Esta metodología es equivalente al cómputo de las componentes principales para la matriz $[X]'$, quedándonos con las dos primeras componentes para definir $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$. Si en $[X]$ hay ritmicidad entonces $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ representan dos posibles patrones que existen en las expresiones de los genes, así como el orden

circular subyacente [Pearson, 1901]. La Figura 2.3 ilustra como los eigengenes forman una estructura circular de la que se puede deducir su orden a partir de la posición de cada muestra en esta circunferencia. Para que estos *eigengenes* sean biológicamente interpretables, $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ deben explicar una variabilidad moderada y similar, de tal manera que se mantenga esta estructura circular.

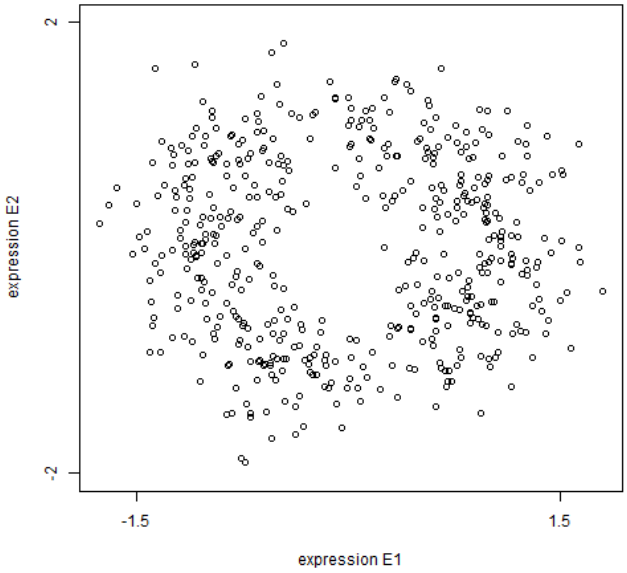


Figura 2.3: Estructura circular formada por $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$.

El orden que define CPCA se obtiene a partir de la transformación que proyecta $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ sobre el círculo unidad:

$$(e_{1,j}, e_{2,j}) = (\frac{E_{1,j}}{\sqrt{E_{1,j}^2 + E_{2,j}^2}}, \frac{E_{2,j}}{\sqrt{E_{1,j}^2 + E_{2,j}^2}}), j = 1, .., m \tag{2.6}$$

Una vez se obtiene la proyección sobre el círculo unidad se calculan las componentes circulares. Para cada par $(e_{1,j}, e_{2,j})$ se obtiene los ángulos $\boldsymbol{\theta} = (\theta_1, ..., \theta_m)$ como:

$$\theta_j = arctan(\frac{e_{1,j}}{e_{2,j}}), j = 1, ..., m \tag{2.7}$$

El orden creciente de estos valores angulares $(\theta_{(1)}, \theta_{(2)}, ..., \theta_{(m)})'$ se corresponde con el vector de ángulos que induce el orden circular que subyace en los datos. Para facilitar la interpretación, estos valores θ_j se pueden escalar al intervalo $[0, 2\pi]$.

2.2.2. Aproximación CPCA mediante red Neuronal

En [Scholz, 2007] se propuso una metodología alternativa para dar solución al problema de la obtención de un orden circular mediante CPCA utilizando un perceptrón multicapa con topología auto-asociativa.

Esta red tiene como entrada un vector de valores de expresión $X_{i,\bullet}$ y como salida un vector $h(X_{i,\bullet})$ en el que se busca que el valor de salida sea similar al de entrada minimizando el error cuadrático medio para la función h. Se dice por tanto que esta red tiene un mapeado a sí misma. En la Figura 2.4 se muestra un ejemplo de una topología auto-asociativa. En la red hay una componente central que realiza una reducción de la dimensión a tamaño $\mathcal{Z}$. A partir de esto, se entiende que la primera mitad de la red comprime o proyecta los valores de $X_{i,\bullet}$ hasta el número de componentes que haya en $\mathcal{Z}$ y posteriormente, la segunda mitad lo vuelve a descomprimir resultando en $h(X_{i,\bullet})$, para que se pueda comparar con $X_{i,\bullet}$ y se realice el ajuste de pesos en la red. También cabe destacar que la capas ocultas W permiten realizar mapeados no lineales de los valores.

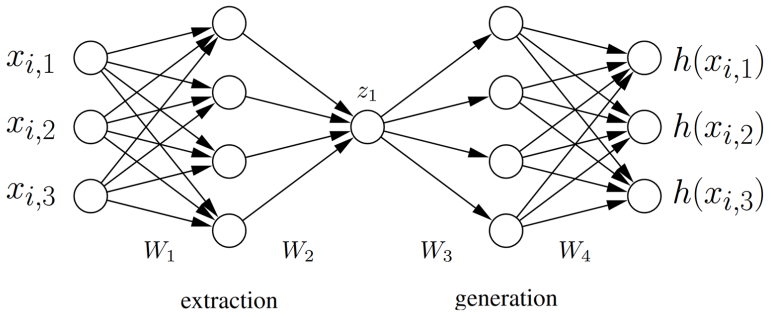


Figura 2.4: Topología de una red auto-asociativa. Modificado de [Scholz, 2007].

A partir de esta estructura se le puede añadir a la componente central de la red una componente circular. Si en $\mathcal{Z}$ se colocan dos unidades, se puede a partir de operaciones trigonométricas, obtener los valores angulares correspondiente al orden circular. Además, en este caso, siguiendo lo expuesto en [Kirby and Miranda, 1996] es importante definir una unidad circular $z_p^2 + z_q^2 = cos(\theta)^2 + \sin(\theta)^2 = 1$ que garantice que las proyecciones están en el círculo unidad. En la Figura 2.5 se muestra un ejemplo de una red auto-asociativa para CPCA. Una vez se ha definido la topología de la red se ajustan los pesos para minimizar el error cuadrático.

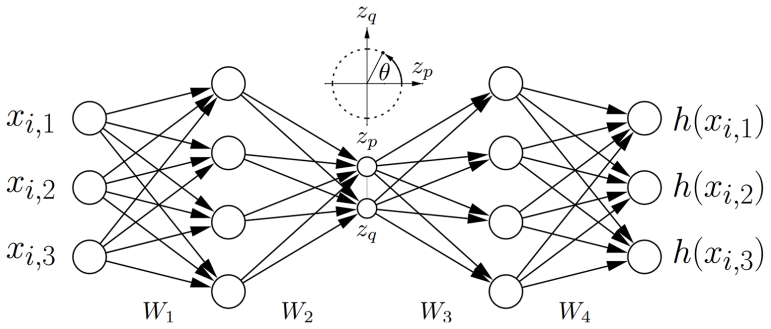


Figura 2.5: Topología de una red auto-asociativa para CPCA. Modificado de [Scholz, 2007].

En la Figura 2.6 se expone un ejemplo del funcionamiento de la red sobre una matriz de genes rítmicos generada artificialmente. La linea roja representa el orden circular obtenido. Esta Figura es bastante

similar a la la Figura 2.3, ambas realizadas con el mismo conjunto de genes, y puesto que ambos métodos persiguen el mismo objetivo solo se diferencian en la manera por la que obtienen dicha reducción de dimensión.

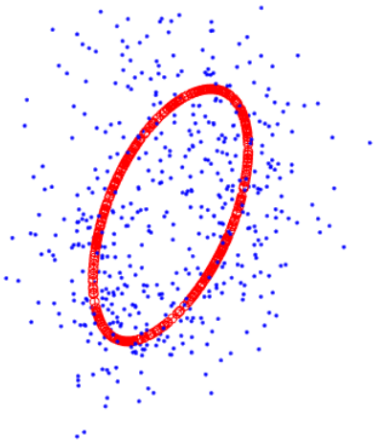


Figura 2.6: Resultados de ajustar unos datos con CPCA.

2.2.3. Aproximación mediante *Travelling Salesman Problem* (TSP)

Este método plantea una aproximación al problema de estimación del orden temporal a partir del TSP [Larriba et al., 2020]. Partiendo de la equivalencia de señal circular en ambos espacios, la expresión de cada gen es posible representarla como un grafo donde los nodos se corresponden con las distintas muestras y las aristas con una distancia entre los valores de expresión del gen en los distintos nodos. De forma que la solución a este problema será la ruta de mínima distancia que recorra todos los nodos empezando y acabando en el mismo punto, es decir un orden circular, que minimice la distancia de todos los genes a dicho orden. Uno de los inconvenientes de este método es que está restringido a una configuración en la que las muestras ordenadas estén equiespaciadas.

2.3. Análisis de señales oscilatorias

Sea $X(t)$ el valor de expresión de un determinado gen en el tiempo t donde $t \in [0, 2\pi]$. Los modelos para señales oscilatorias se definen como un modelo clásico de señal más ruido:

$$X(t) = \mu(t) + e(t) \tag{2.8}$$

Donde $\mu(t)$ es una señal oscilatoria y $e(t)$ representa el error en el tiempo t.

El análisis de señales oscilatorias se puede abordar desde la perspectiva del análisis funcional, donde cada dato se define como una función. Así, $X(t)$ representa los datos observados a lo largo del tiempo $[0, 2\pi]$ para los que subyace una señal oscilatoria $\mu(t)$.

Esta expresión también se puede trabajar como un dato funcional, es decir, aquellos datos en los que se define cada observación a partir de una función. Por tanto, cada dato funcional para señales oscilatorias

está definido por $X(t)$, la señal circular y los diferentes instantes t en los que ocurre $X(t)$.

En esta sección se introduce el modelo Frequency Modulated Möbious (FMM), un modelo para señales oscilatorias que puede utilizarse para ajustar el patrón de expresión de los genes. Se define también el modelo Cosinor, que podría considerarse un caso particular del modelo FMM, y que ha sido ampliamente utilizado en la literatura para resolver el problema de la ritmicidad en la expresión génica. Por último, se describe el análisis de componentes principales funcionales, punto de partida para realizar el agrupamiento de los genes considerados rítmicos en diferentes patrones característicos.

2.3.1. Modelo para señales oscilatorias

Cosinor

El modelo Cosinor fue propuesto por primera vez en [Cornelissen, 2014], mediante una formulación matemática de ondas sinusoidales. La definición del modelo es la siguiente:

$$\begin{aligned} X(t) &= \mu(t) + e(t) \\ &= M + A\cos(t+\phi) + e(t), t \in [0, 2\pi] \end{aligned} \tag{2.9}$$

Donde M es el valor medio de la onda, también conocido como *Midline Statistic Of Rhythm* (MESOR), A es la amplitud y ϕ es la acrofase, es decir, el tiempo transcurrido desde el inicio de la onda hasta el valor máximo. El principal inconveniente de este modelo es que solo permite describir ondas simétricas

Frequency Modulated Möbius: FMM

El modelo fue propuesto por primera vez en [Rueda et al., 2019] con una formulación matemática en la que aparecen parámetros de forma de la onda, lo que permite describir una variedad mayor de patrones, y no solo ondas simétricas.

Se define el modelo FMM para $X(t)$ como:

$$\begin{aligned} X(t) &= \mu(t) + e(t) \\ &= M + A\cos(\varphi(t)) + e(t), t \in [0, 2\pi] \end{aligned} \tag{2.10}$$

Donde:

- $M \in R, A \in R^+$
- $\varphi(t) = \beta + 2\arctan(\omega\tan(\frac{t-\alpha}{2})); \alpha, \beta \in [0, 2\pi]; \omega \in [0, 1]$
- $e(t) \sim\ N(0, \sigma^2)$

Estos cinco parámetros describen los distintos aspectos de la onda:

- M: es el *intercept* del modelo.
- A: representa la amplitud de la onda.
- α: describe la localización de la fase de la onda.
- β y ω: son parámetros de la forma de la onda. β representa la simetría de la onda. Además indica si la onda tiene una estructura *up-down* o *down-up* para sus instantes de máxima expresión. Cuando β vale 0 o π la onda es simétrica. ω indica el apuntamiento (anchura) de la onda, $\omega = 0$ indica una onda muy apuntada mientras que $\omega = 1$ indica una onda sinusoidal.

La flexibilidad del modelo FMM viene dada por el término $\varphi(t)$. Esta variable angular representa el movimiento periódico de la oscilación a partir de tres parámetros lineales y angulares, a diferencia del modelo Cosinor donde este término se corresponde con una variable lineal.

En la Figura 2.7 se comparan curvas FMM para distintos parámetros de β y ω para ver la variedad de patrones al variar los dos parámetros de forma.

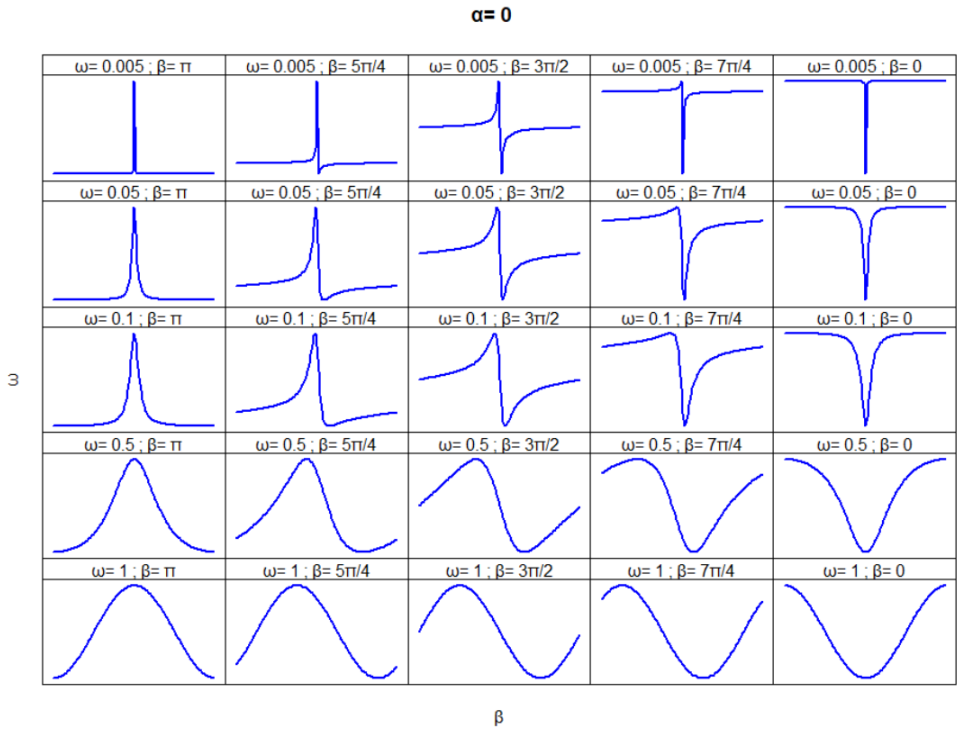


Figura 2.7: Modelos FMM para M=0, $A = 1$ y $\alpha = 0$ variando los valores de β y ω . Fuente [Rueda et al., 2019].

Otros parámetros importantes en este contexto que se pueden derivar del modelo FMM son los instantes en los que el modelo alcanza el valor máximo y mínimo de expresión, t_U y t_L respectivamente.

$$t_U = \alpha + 2\arctan(\frac{1}{\omega}\tan(\frac{-\beta}{2})) \tag{2.11}$$

$$t_L = \alpha + 2\arctan(\frac{1}{\omega}\tan(\frac{\pi-\beta}{2})) \tag{2.12}$$

Medida de calidad del modelo

Para medir la bondad de ajuste del modelo se utiliza el coeficiente de determinación R^2. Este coeficiente está entre 0 y 1 indicando un mejor ajuste cuanto más próximo esté a 1. Se define el R^2 para una onda como:

$$R^2 = 1 - \frac{\sum_t (X(t) - \hat{\mu}(t))^2}{\sum_t (X(t) - \bar{X})^2} \tag{2.13}$$

Donde $\hat{\mu}(t)$ es la señal circular predicha para t, $X(t)$ es la observación en el momento t y $\bar{X}$ es el valor medio de X.

2.3.2. Análisis de componentes principales funcionales

El Análisis de Componentes Principales Funcionales (FPCA) se define como un PCA donde los datos de entrada se corresponden con datos funcionales. Las componentes obtenidas son por tanto ortogonales entre sí al igual que en PCA. Sin embargo, para ajustar los datos funcionales primero se resumen los valores $X(t)$ en una función conocida que describa su forma. Generalmente, se usan ***B-splines*** para esta definición, ya que facilitan el cálculo del FPCA al facilitar el cálculo integral [He et al., 2022].

A partir de FPCA se puede realizar un *clustering* utilizando como datos de entrada al algoritmo un número de componentes de este análisis.

2.4. Enriquecimiento funcional

El objetivo del enriquecimiento funcional es estudiar si hay relación entre conjuntos de genes previamente definidos en bases de datos y conjuntos de genes que se obtienen como resultado de un estudio gen a gen. En el análisis de enriquecimiento se busca encontrar qué clases, entre las definidas en bases de datos que recogen el conocimiento biológico disponible, de genes están sobre-representadas e infra-representadas en el conjunto de genes propuesto.

Gene Set Enrichment Analysis (GSEA) es uno de los método más populares para realizar el enriquecimiento [Subramanian et al., 2005]. El método calcula si los genes de L están sobre o infra representados en un conjunto C siendo $L = g_1, ..g_N$ una lista de genes ordenados en función de un fenotipo $\boldsymbol{F} = f_1, .., f_N$ y C un conjunto de genes para una categoría derivada de manera independiente, como algún término definido en la base de datos GO. En este problema, la definición de fenotipo se usa de manera amplia, ya que puede ser cualquier medida numérica que permita ordenar los genes de L según su importancia, por ejemplo se pueden usar medidas de la bondad de ajuste de un modelo, como el R^2.
Los pasos del método son los siguientes:

1. Calculo del *Enrichment Score (ES): ES* es una medida del grado en el que el conjunto C está sobre o infra representado en L. Para calcularlo se recorre la lista L incrementando el estadístico cuando el gen pertenece al conjunto C y decrementando cuando el gen no pertenece a C. Se definen $h_C(i)$ como la función que incrementa su magnitud cuando el gen i pertenece al conjunto C:

$$h_C(i) = \frac{1}{\sum_{g_i \in C} |f_i|^p} \sum_{\substack{g_i \in C \\ i' \leq i}} |f_{i'}|^p \tag{2.14}$$

siendo p un valor entre 0 y 1 que representa la importancia que se le dan a los valores fenotípicos. Se define también $m_C(i)$ como la función empírica de los genes que no están en C.

$$m_C(i) = \sum_{\substack{g_i \notin C \\ i' \leq i}} \frac{1}{\sum_{g_i \notin C} 1} \tag{2.15}$$

El estadístico *ES* es la máxima desviación entre las dos funciones que se observa al recorrer L:

$$ES_C = \max_{1 \leq i \leq N} |h_C(i) - m_C(i)| \tag{2.16}$$

Por tanto el estadístico *ES* se corresponde a un estadístico de Kolmogorov-Smirnov con pesos [Hollander et al., 1999].

2. Estimación del nivel de significación de *ES*: Se genera la distribución nula del *ES* mediante un contraste de permutaciones. Por cada permutación de los valores fenotípicos, se reordena L y se calcula el nuevo ES^b. Considerando un numero B grande de permutaciones, los valores de los ES^b simulan la distribución nula del estadístico *ES*. El p-valor para cada C se calcula como la proporción de permutaciones en las que el valor ES^b es mayor que *ES*.

3. Ajuste de los p-valores: se ajustan los p-valores para tener en cuenta el problema de las comparaciones múltiples. Frecuentemente, para ello se estima la tasa de falsos descubrimientos (FDR).

2.5. Algoritmo para el análisis de genes rítmicos

A continuación se explican las etapas propuestas en el trabajo para el análisis de una matriz de datos de expresión desordenados para un tejido. Una esquematización del algoritmo puede verse en la Figura 2.8:

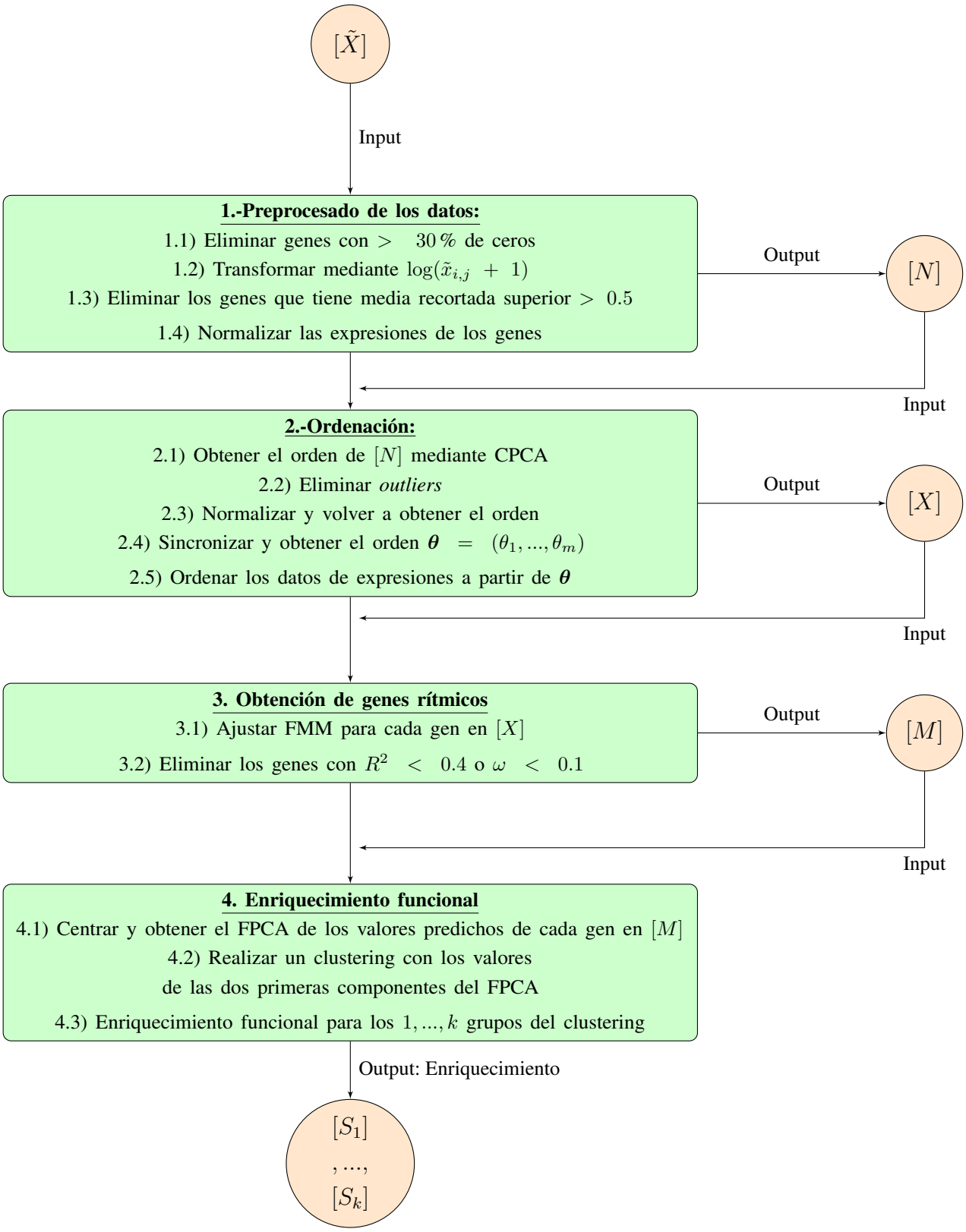


Figura 2.8: Esquema del algoritmo

2.5.1. Preprocesado de los datos

Esta etapa está inspirada en la metodología de [Larriba et al., 2022] y contiene 4 pasos. La entrada a esta etapa es la matriz $[\tilde{X}]$ de datos de expresiones crudos y desordenados para un tejido.

1. Se eliminan los genes que aportan poca información, ya que tienen demasiados valores a 0. Se eliminan los genes que tengan más del 30 % de ceros en su expresión.

2. Se transforman los datos mediante la transformación $\log_2(\tilde{x}_{i,j} + 1)$. Un problema que tienen los datos de RNA-seq es que, en general, tienen una distribución muy asimétrica a la derecha y no son homocedásticos. La transformación logarítmica ayudará a solucionar estos problemas, así como a reducir el número de *outliers* en los datos. En el contexto de los datos de expresión es habitual utilizar el logaritmo en base 2 por una cuestión de interpretación. Además se ha de sumar una constante a los datos de expresión para evitar los posibles ceros.

3. Se busca eliminar los genes con niveles de expresión bajos. Para ello se usa el valor de una media recortada superiormente: se ignoran el 10 % de las muestras con valores de expresión mayores en el gen. Esto permite evitar el efecto que tienen los *outliers* en la expresión media. Se eliminan los genes cuya media sea menor que 0.5. Los detalles de este paso se encuentran en la sección 5.1 del Anexo. En [Ruben et al., 2018] se propone una lista de 54 genes rítmicos. En este paso se usan estos genes para validar que no se eliminan demasiados genes del tejido mediante este filtro. No todos estos genes tienen porque ser rítmicos en todos los tejidos, pero si debería de haber una proporción alta que lo sean para cada tejido.

4. Se normalizan los valores de cada gen entre $[-1, 1]$ utilizando la normalización min-max.

Se obtiene como output la matriz $[N]$ de datos de expresiones normalizadas y procesadas.

2.5.2. Ordenación

La entrada a esta etapa es una matriz de datos de expresiones génicas procesada y normalizada $[N]$. El proceso es similar al aplicado en [Larriba et al., 2022]. Esta etapa consta de 5 pasos.

1. Se obtiene un orden para las muestras a partir de los 12 genes *core*: *PER1, PER2, PER3, CRY1, CRY2, ARNTl, CLOCK, NR1D1, RORA, DBP, TEF, STAT3* . Estos genes generalmente tiene patrones rítmicos en la mayoría de tejidos humanos, como se observó en [Zhang et al., 2014] y han sido considerados anteriormente en otros análisis de los ritmos circadianos [Anafi et al., 2017, Ruben et al., 2018, Wu et al., 2018, Larriba et al., 2020, Larriba et al., 2022].

   Para la sub-matriz de $[N]$ de estos 12 genes se calculan las componentes principales circulares mediante el método expuesto en la sección 2.2.1. Se obtiene un primer orden con el que se ordena la matriz completa de datos.

2. Se eliminan los *outliers*. Para esto, se ajusta un modelo FMM para cada uno de los 12 genes *core* indicados anteriormente. A partir de los residuos de estos modelos se buscan los *outliers* en los datos. Para detectar los *outliers* de un gen se usan los residuos estandarizados. La definición de los residuos

estandarizados puede encontrarse en la sección 5.2 del anexo. Se consideran *outliers* aquellos que tengan residuos estandarizados fuera del intervalo $[-3, 3]$. Se eliminan aquellas muestras que sean *outliers* en al menos uno de los doce genes *core*.

3. Se vuelve a normalizar la matriz de datos ya que eliminar los *outliers* puede llevar a que cambien los extremos en la normalización min-max. Se obtiene de nuevo mediante CPCA y a partir de los 12 genes *core* el orden con el que ordenar todos los datos.

4. Se sincronizan los genes. Es importante tener en cuenta que el orden obtenido es un orden circular, por lo que existen 2m posibles configuraciones para el orden. Hay m puntos posibles de comienzo y dos posibles direcciones para el orden. Para escoger el punto de inicio del orden se toman las siguientes decisiones. Considerando el ajuste del modelo FMM para el gen *ARNTL*, se fija el punto de máxima expresión para que este esté en π. Se sabe que este gen tiene su momento de máxima expresión antes del periodo inactivo [Ruben et al., 2018], lo que divide el intervalo en el periodo activo e inactivo. Para decidir la dirección se utilizan también los ajustes del modelo FMM a los genes *PER1, PER2, PER3, CRY1, CRY2* que se espera que tengan su máxima expresión en el periodo activo [Korenčič et al., 2014, Mure et al., 2018]. De esta manera el intervalo $[0, \pi]$ representa el periodo de actividad. Sincronizar los genes permite comparar los resultados con otros obtenidos previamente, así como facilitar la detección de patrones.

5. Se ordenan los datos a partir del orden obtenido y sincronizado.

Se obtiene como output la matriz [X] de expresiones de genes ordenadas.

2.5.3. Obtención de genes rítmicos

En esta etapa se seleccionan los genes que son rítmicos. La entrada de la etapa es la matriz [X] de expresiones de genes ordenadas.

1. Se ajusta un modelo FMM para cada uno de los genes en $[X]$.

2. Se eliminan los genes que tengan $R^2 < 0.4$ o $\omega < 0.1$. Se eliminan estos genes para descartar aquellos con excesivo ruido o apuntados al tratarse de genes poco informativos.

La salida de la etapa es la matriz $[M]$ de valores predichos de los genes a partir del modelo FMM.

2.5.4. Enriquecimiento funcional

La entrada a esta etapa es la matriz $[M]$. El objetivo es clasificar los genes en grupos de acuerdo a su similitud entre patrones y el estudio de las principales funciones biológicas en las que los genes de cada grupo están involucrados.

1. Se centran los genes respecto a la media, con el objetivo de que genes que tienen el mismo patrón pero con distintos niveles de expresión sean clasificados en el mismo patrón. Se ajustan unas componentes principales funcionales para los genes en [M].

2. Se realiza un agrupamiento de los genes a partir del procedimiento de *clustering* K-Medoids, seleccionando el número óptimo k de *clusters* a partir del algoritmo establecido en la sección 5.3 del Anexo, usando como datos de entrada al algoritmo las dos primeras componentes del FPCA. Para cada *cluster* se calcula su patrón medio a partir de los parámetros de cada gen en $[M]$, y se promedian teniendo en cuenta el valor R^2 de cada gen como peso para cada parámetro. Los parámetros α y β son circulares por lo que se debe utilizar la media con pesos expuesta en la sección 2.1.1.

3. Se realiza el enriquecimiento funcional mediante el método GSEA para cada *cluster*.

La salidas de esta etapa son los resultados del enriquecimiento para cada *cluster* $[S_1], ..., [S_k]$ que aportarán información sobre la funcionalidad biológica.

Resultados

Esta sección compara primero los diferentes métodos para resolver el problema de la estimación del orden temporal. También presenta descrito en la sección 3.2 el análisis para uno de los tejidos de GTEX.

3.1. Simulación

El objetivo de este estudio de simulación es comparar la eficiencia de los tres métodos que resuelven el problema del orden temporal propuestos en la sección 2.2. Para estudiar estas diferencias, se simulan datos de expresión a partir de 4 patrones de señales oscilatorias generados con el modelo FMM, tanto simétricos como asimétricos. La configuración de los parámetros FMM aparecen el la Tabla 3.1. Para imitar lo que ocurre en los estudios de genéticos, donde en general el número de genes no rítmicos es mayor que el de rítmicos, también se han obtenido datos a partir de 6 patrones planos (no rítmicos). Los datos se han simulado a lo largo de 500 observaciones, que posteriormente se han desordenado, de acuerdo a dos configuraciones de ruido $\sigma^2 = 0.05$ y $\sigma^2 = 0.01$

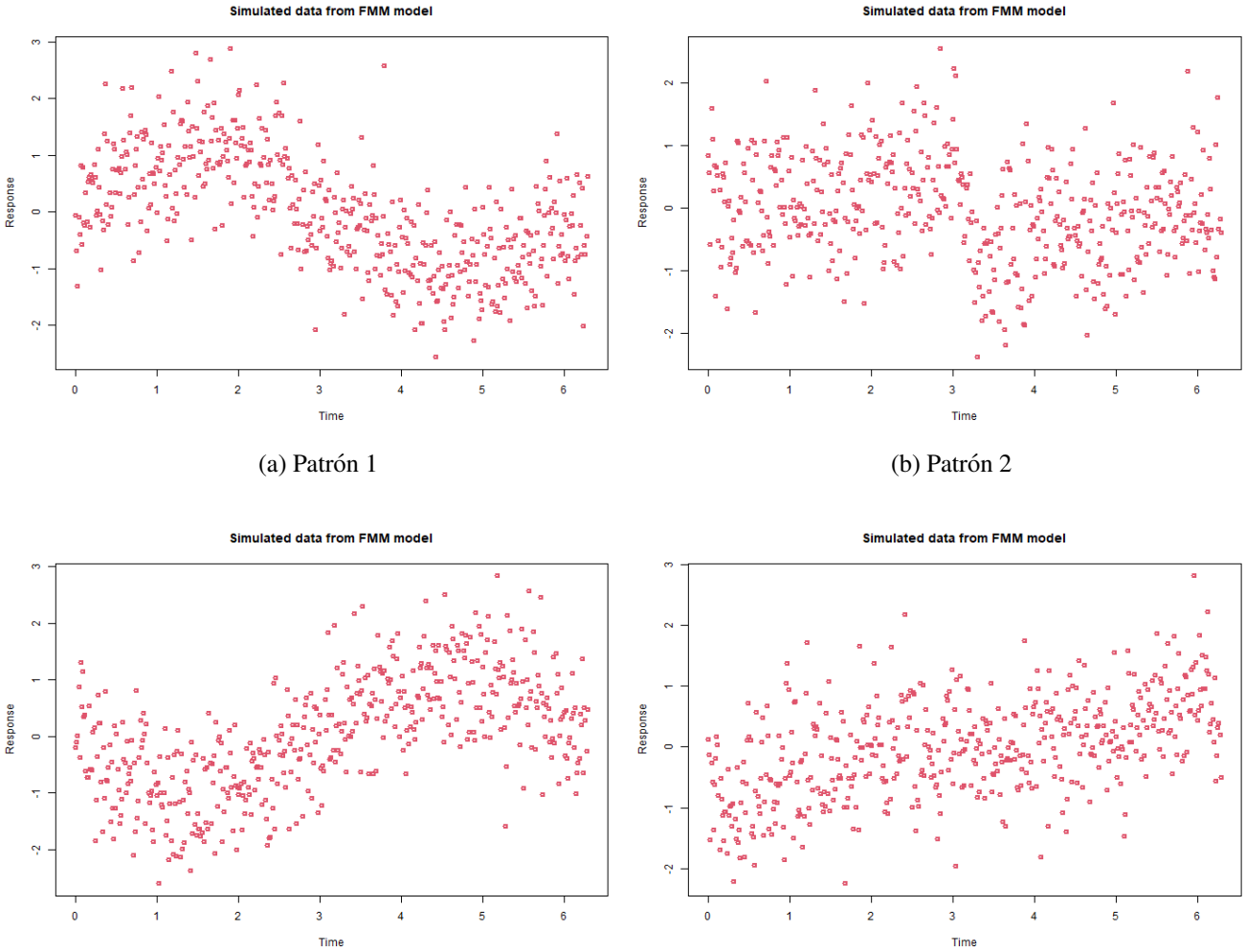
patrón	M	A	α	β	ω
1	0	1	0	$3\pi/2$	0.9
2	0	1	0	$3\pi/2$	0.1
3	0	1	$\pi/2$	$3\pi/2$	0.9
4	0	1	π	$3\pi/2$	0.1

Tabla 3.1: Parámetros para las señales oscilatorias usadas en la simulación.

La Figura 3.1 ilustra los patrones rítmicos simulados para $\sigma^2 =$ Este experimento se repite 10 veces para cada configuración del ruido.

Para poder comparar los órdenes obtenidos por cada método se deben sincronizar los patrones. Para esto se sincroniza el punto de máxima expresión del patrón 3 con el punto máximo de esta patrón en el orden real y para decidir la dirección de rotación se decide que el punto de máxima expresión del patrón 1 debe ocurrir en el intervalo $[0, \pi]$.

A continuación se describen algunas particularidades sobre los resultados de la simulación para cada uno de estos métodos, y se concluye con una comparación en términos de una medida de similitud ente órdenes.



(a) Patrón 1

(b) Patrón 2

(c) Patrón 3

(d) Patrón 4

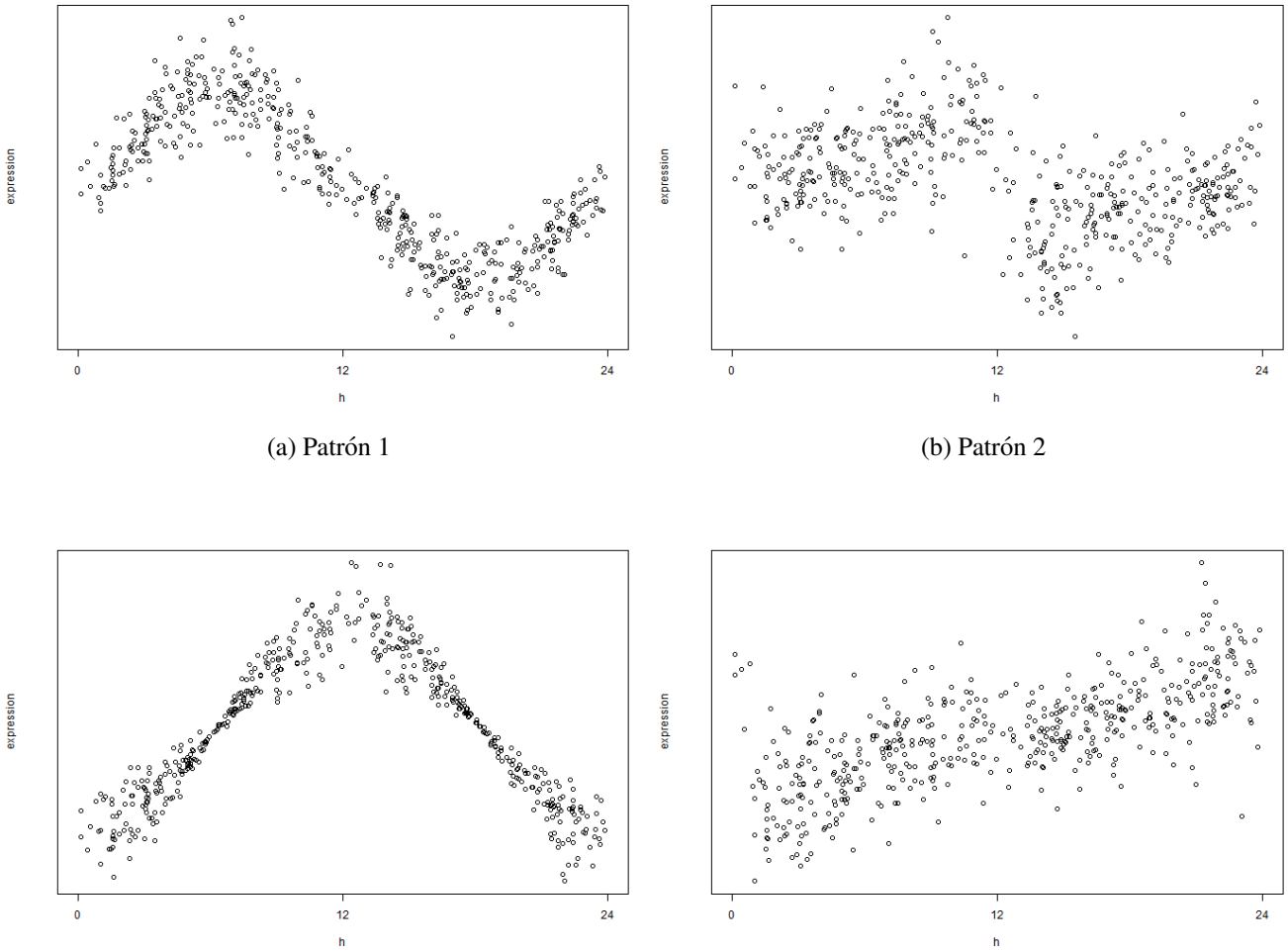
Figura 3.1: Patrones simulados a partir de los parámetros de la Tabla 3.1. con $\sigma^2 = 0.1$.

3.1.1. Simulación: CPCA

La Figura 3.2 ilustra el comportamiento del procedimiento CPCA en una de las repeticiones. Como se comentó, es importante, para que los resultados sean biológicamente interpretables, que los dos primeros *eigengenes* $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ expliquen una variabilidad moderada entre ambos, tal y como se observa en la Tabla 3.2

σ^2	*Eigengene*	% de variabilidad explicada
0.05	$\boldsymbol{E_1}$	36.867
0.05	$\boldsymbol{E_2}$	31.081
0.1	$\boldsymbol{E_1}$	31.299
0.1	$\boldsymbol{E_2}$	26.948

Tabla 3.2: Variabilidad explicada por los *eigengenes* en ambos escenarios.



(a) Patrón 1

(b) Patrón 2

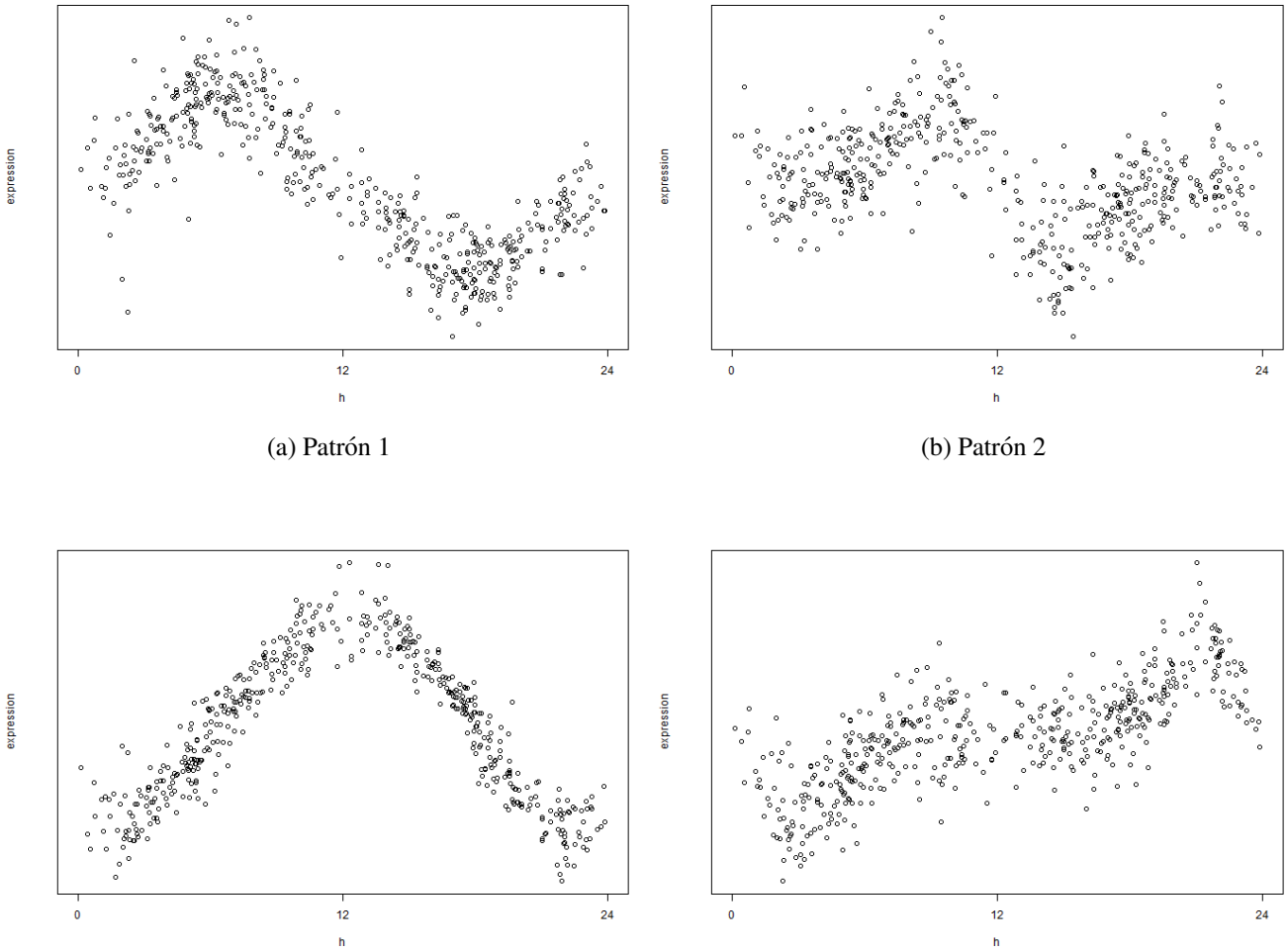
(c) Patrón 3

(d) Patrón 4

Figura 3.2: Ilustración de una reconstrucción en los 4 patrones rítmicos mediante el método CPCA.

3.1.2. Simulación: Red neuronal

La Figura 3.3 ilustra el comportamiento del método Red neuronal en una de las repeticiones. En general se observa mayores diferencias con el métodos CPCA en los patrones asimétricos y una variabilidad similar en el patrón reconstruido.



(a) Patrón 1

(b) Patrón 2

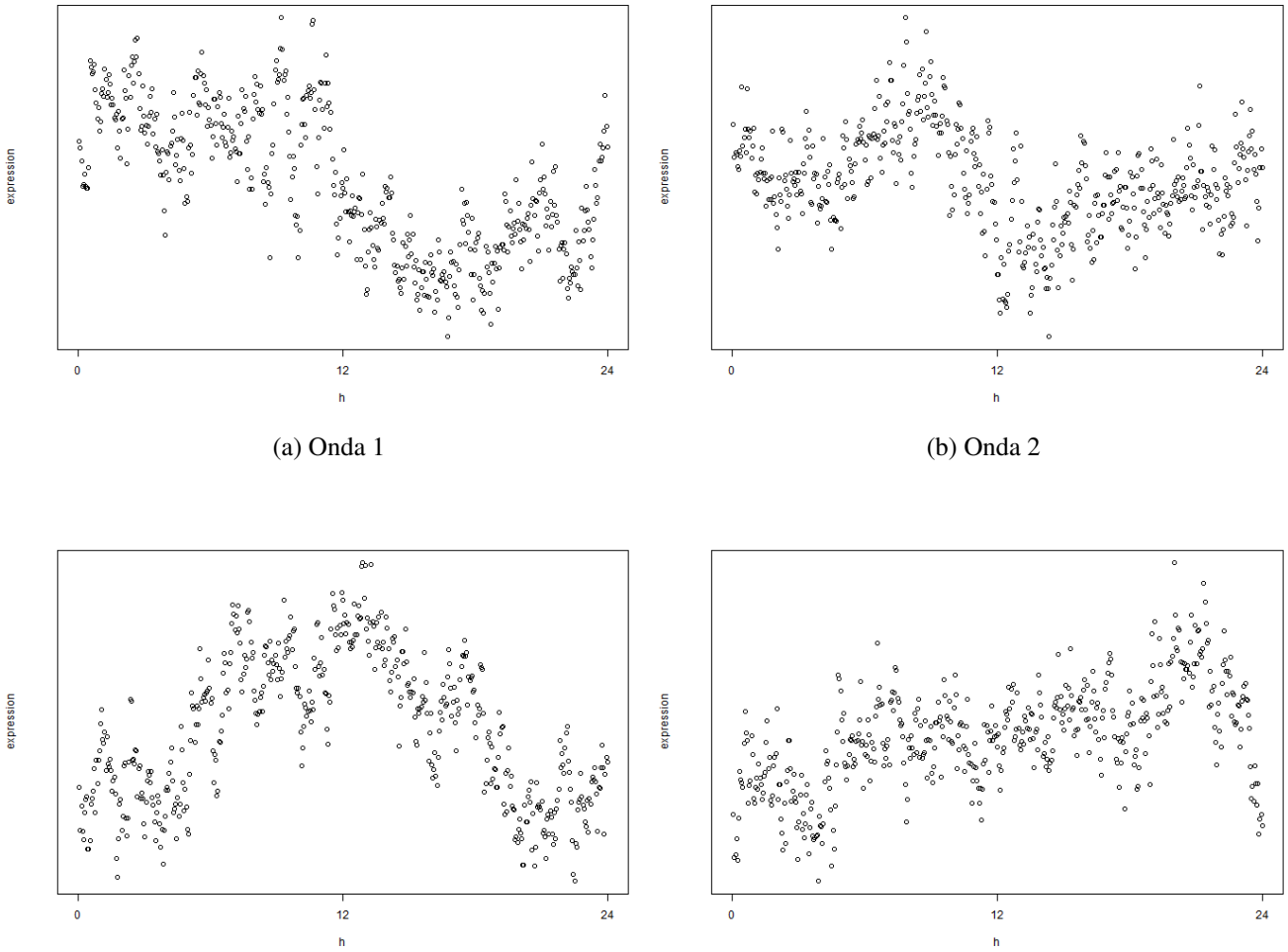
(c) Patrón 3

(d) Patrón 4

Figura 3.3: Ilustración de una reconstrucción en los 4 patrones rítmicos mediante el método de la red neuronal.

3.1.3. Simulación: TSP

La Figura 3.4, ilustra el comportamiento de la ordenación por TSP en una de las repeticiones. En general se observan patrones más irregulares que en los métodos anteriores. Este método tiene además la desventaja de que es computacionalmente más intensivo que los otros dos, debido a las distintas heurísticas que tiene que realizar para tratar de encontrar el óptimo. Este método solo permite obtener observaciones equiespaciadas y dada la naturaleza del problema, queda claro que es importante que el método obtenga el vector de ángulos $\boldsymbol{\theta}$, ya que para datos de donantes no están equiespacidas. Por este motivo se descarta usar TSP para obtener el orden de las observaciones.



(a) Onda 1

(b) Onda 2

(c) Onda 3

(d) Onda 4

Figura 3.4: Ilustración de una reconstrucción en los 4 patrones rítmicos mediante el método TSP.

3.1.4. Comparación y selección del mejor método

A la vista de los resultados anteriores se procede a decidir qué método de ordenación es el más eficiente en términos de la siguiente métrica que cuantifica la discrepancia entre dos órdenes:

$$\sum_{i=i}^{n}\sum_{j=1}^{m}\frac{|x_{i,j}-\mu_{i,j}|}{n} \tag{3.1}$$

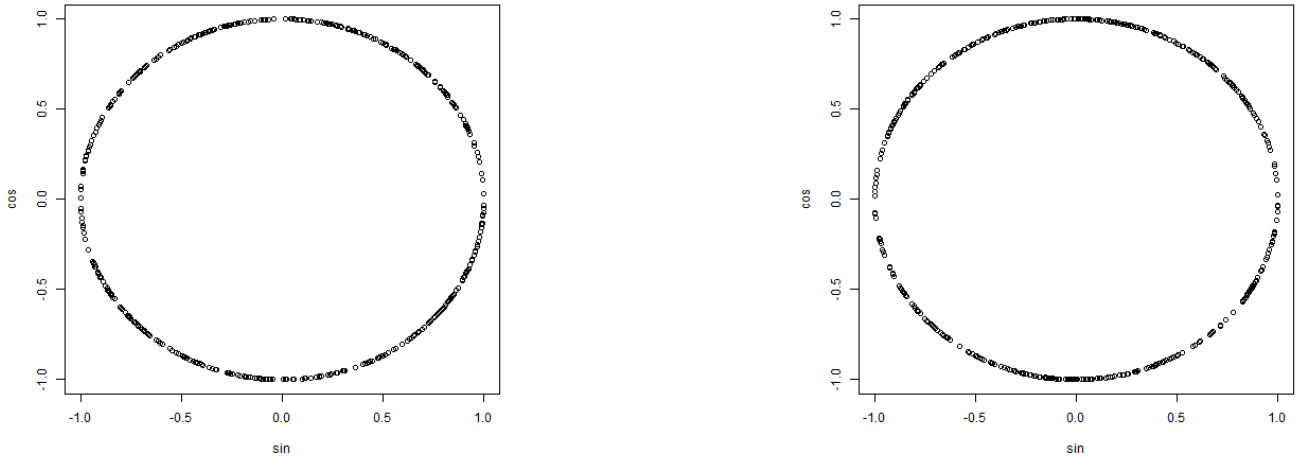
Siendo $\mu_{i,j}$ el valor del i-ésimo patrón FMM en el instante j, $i = 1, ..., 4$ $j = 1, ..., 500$. Esta métrica toma valores positivos y será mejor cuanto menor sea su valor.

Las métricas para los distintos métodos se pueden ver en la Tabla 3.3.

σ^2	CPCA *Eigengenes*	Red Neuronal
0.05	0.1836 ± 0.003	0.1859 ± 0.0061
0.1	0.2636 ± 0.0063	0.2645 ± 0.0081

Tabla 3.3: Valores (media $\pm$ desviación típica) para la métrica 3.1 para cada método y escenario a partir de 10 repeticiones.

A partir de los resultados de la Tabla 3.3 se concluye que ambos métodos se comportan de manera similar. Un aspecto importante para la validez de los resultados obtenidos para este problema es que la reconstrucción obtenida sea consistente con la supoción de que se disponen de muestras a lo largo de todo el periodo. Esto se traduce en que el orden circular estimado se distribuya de manera uniforme a lo largo del círculo unidad, tal y como se observa en la Figura 3.5 para ambos métodos.



(a) Recubrimiento CPCA

(b) Recubrimiento Red Neuronal

Figura 3.5: Comparación del recubrimiento entre el método CPCA y Red Neuronal.

Una de las desventajas de la red neuronal es que es un algoritmo iterativo que en caso de escoger un número incorrecto de iteraciones el método puede llevar a sobreajuste, y por tanto los resultados dejan de tener interpretación biológica. En la Figura 3.6 se puede ver un ejemplo de los resultados cuando se utiliza un número demasiado alto de iteraciones. Esto lleva a la necesidad de ajustar el número de iteraciones como un hiperparametro que se debe estudiar antes de realizar la estimación del orden. Esto añade complejidad al procedimiento mientras que los resultados mostrados en la Tabla 3.3 no indican que esta complejidad aporte una mejora que la justifique.

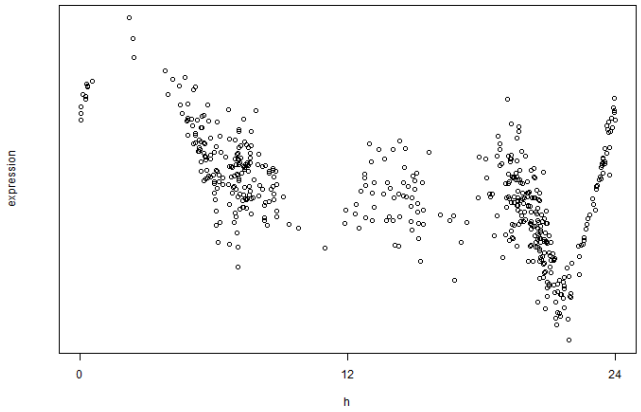


Figura 3.6: Resultados obtenidos cuando se sobreajusta la red para el patrón 4.

Por tanto, en adelante en este trabajo se considerará el método CPCA para abordar el problema de estimación del orden temporal.

3.2. Análisis de los datos de GTEX

Se procede a continuación a estudiar los datos de la base de datos GTEX. En primer lugar, se describe la base de datos, y posteriormente se presentan los resultados de aplicar el algoritmo propuesto en la sección para el tejido muscular de la base GTEX.

3.2.1. Descripción de los datos

Los datos estudiados provienen de la base de datos GTEX. La base de datos tiene 948 donantes. Se miden expresiones de genes de cada donante en varios tejidos, por lo que la matriz de datos tiene una dimensión de 56200 genes y 17382 muestras. Para cada donante se conoce además varias variables fenotípicas que son de interés en este trabajo y se describen a continuación:

- TOD, hora de muerte del donante, desde las 0:00 a las 23:59.
- Sexo del donante, 1=Hombre, 2=Mujer.
- Edad del donante codificada como variable categórica en 6 grupos. Los grupos de edades abarcan edades entre 20 y 80, en intervalos de 10 años.

Para los tejidos existen dos identificadores diferentes, uno más general, por ejemplo el identificador *Muscle* representa todas las muestras provenientes de músculos, y un identificador más especifico, como por ejemplo *Muscle-Skeletal*, representa las muestras que son de tejido músculo esquelético. En este trabajo se va a utilizar el identificador específico. En la Tabla 3.4 se puede ver el número de muestras para los tejidos con más de 600 muestras.

Tejido	nº de observaciones
Muscle-Skeletal	803
Whole Blood	755
Skin - Sun Exposed (Lower Leg)	701
Adipose -Subcutaneous	663
Artery - Tibial	663
Thyroid	653
Nerve - Tibial	619
Sin - Not Sun Exposed (Suprapubic)	604

Tabla 3.4: Tejidos con más de 600 observaciones

3.2.2. Resultados de las distintas etapas del algoritmo en el tejido músculo esquelético

A continuación se aplica el algoritmo descrito en la sección 2.5 al tejido *muscle skeletal*, de aquí en adelante tejido músculo esquelético. Según lo expuesto en [Ruben et al., 2018] y en las referencias del mismo, se caracteriza por ser uno de los más rítmicos en humanos.

En este trabajo se analizan 803 muestras correspondientes a datos de expresión post-mortem para el tejido músculo esquelético. La Tabla 3.5 describe la distribución de estas muestras atendiendo a las variables fenotípicas sexo y edad.

	Frecuencia	Proporción	IC
Sexo: Hombre	543	0.6762	$[0.6438, 0.7086]$
Sexo: Mujer	260	0.3238	$[0.2914, 0.3562]$
Edad: 20-29	67	0.0834	$[0.0643, 0.1026]$
Edad: 30-39	65	0.0809	$[0.0621, 0.0998]$
Edad: 40-49	124	0.1544	$[0.1294, 0.1794]$
Edad: 50-59	255	0.3176	$[0.2854, 0.3498]$
Edad: 60-69	264	0.3288	$[0.2963, 0.3613]$
Edad: 70-79	28	0.0349	$[0.0222, 0.0476]$

Tabla 3.5: Frecuencia, proporción e Intervalo de Confianza (IC) para la proporción de la distribución de muestras en músculo esquelético por sexo y grupo de edad.

Como se ha indicado, una de las suposiciones es que disponga de datos a lo largo de todo el periodo (24h). Esta suposición es cierta para el TOD proporcionado por GTEX, observando un porcentaje de 22.5 % valores ausentes. En cualquier caso, como se ha indicado, la estimación del TOD puede no ser del todo precisa. En la Figura 3.7 se transforman los TODs al intervalo $[0, 2\pi]$ y se muestran los resultados del recubrimiento del círculo unidad mediante las operaciones seno y coseno.

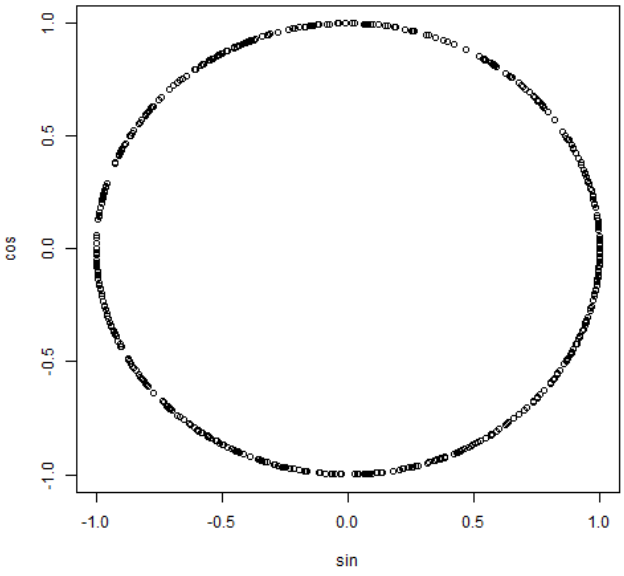


Figura 3.7: Distribución de las horas de muerte a lo largo del día.

A continuación se ilustran los resultados para cada paso del algoritmo en el tejido músculo esquelético

Preprocesado de los datos

Se eliminan los genes que tienen una proporción de ceros mayor que 0.3. En la Figura 3.8 se expone un ejemplo de un gen eliminado en esta parte. Con este filtro se eliminan 33469 genes, es decir una proporción de 0.5955 (IC $[0.5915, 0.5996]$) de los 56200 genes de la base de datos.

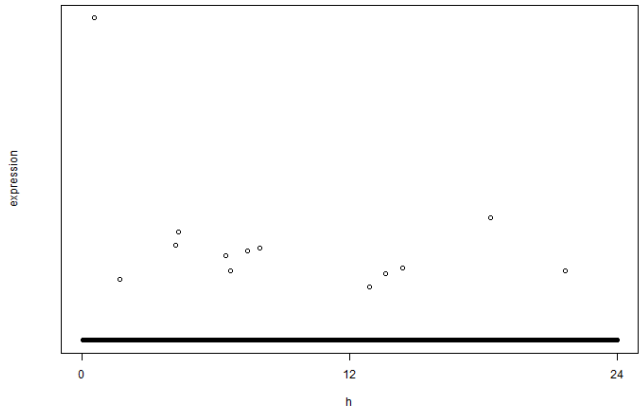
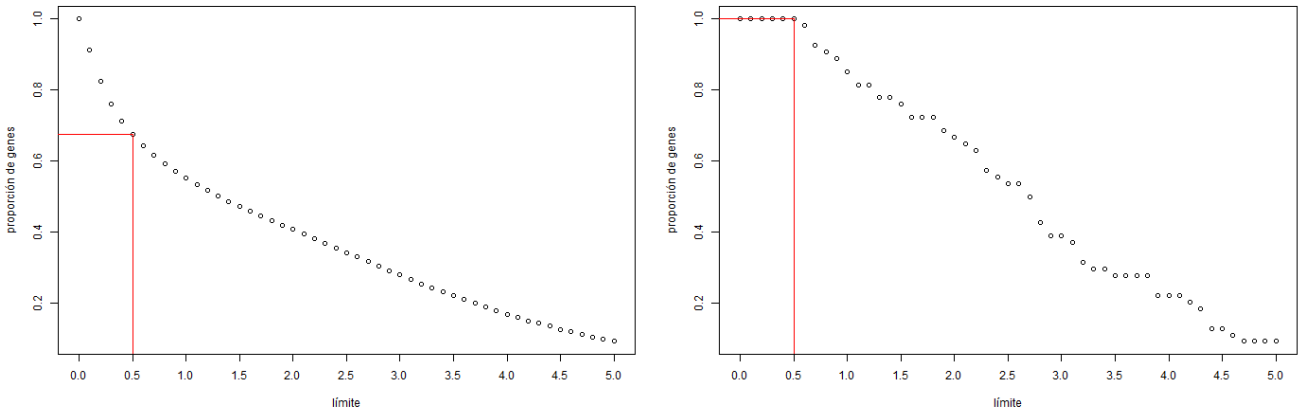


Figura 3.8: Gen eliminado por tener demasiados valores de expresión a cero.

Se eliminan los genes que tienen muchos valores cercanos a 0 por tratarse de genes poco expresados. A partir del valor de la media recortada se estudia si el límite 0.5 es apropiado para no perder ninguno de los 54 genes rítmicos. En la figura 3.9 se representa la proporción de genes que se no se eliminan, para el total de genes y para el conjunto de 54 genes, en función del valor límite.



(a) Proporción de genes del total que se seleccionan.

(b) Proporción de genes entre los 54 que se seleccionan.

Figura 3.9: Proporción de genes que no son eliminados por la media recortada para distintos valores límite.

De acuerdo con lo anterior, si se eliminan los genes con media menor que 0.5, se elimina una proporción de 0.3264 (IC $[0.3203, 0.3325]$) de los 22731 genes, sin eliminar ninguno de los 54 genes rítmicos. En la Figura 3.10 se expone un ejemplo de un gen eliminado cuya expresión está muy cercana al cero. Una vez se han eliminado estos genes se normalizan los datos entre $[-1, 1]$.

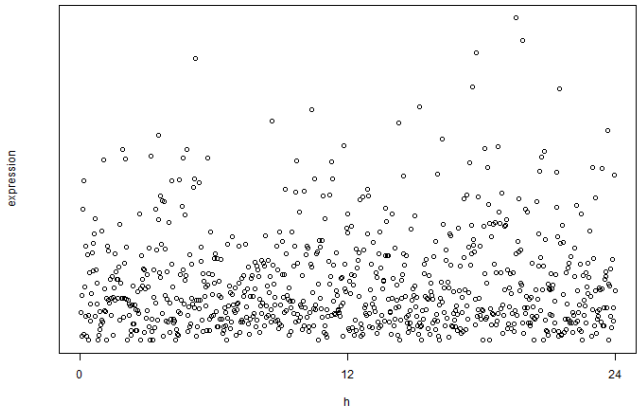


Figura 3.10: Gen eliminado por tener una expresion media cercana a cero.

Al final de esta etapa se ha reducido el número de genes potencialmente rítmicos a 15312. Es importante que la cantidad de genes al final de esta etapa se haya reducido considerablemente debido al coste computacional del ajuste de los modelos FMM en las expresiones del gen (una vez ordenadas) a lo largo de más de 700 instantes de tiempo.

Ordenación

Se aplica el procedimiento CPCA a partir de la matriz de los 12 genes *core*. Con el orden obtenido se ajusta el modelo FMM a los *core* para eliminar posibles muestras *outliers*, según lo descrito en la sección 2.5.2. La proporción de *outliers* eliminados es de 0.0734 (IC $[0.0554, 0.0915]$). En la Figura 3.11 se muestran los *outliers* en el gen *TEF*. Los *outliers* marcados en *TEF*, pueden haber sido identificados como tal en cualquiera de los genes core, no solo en TEF.

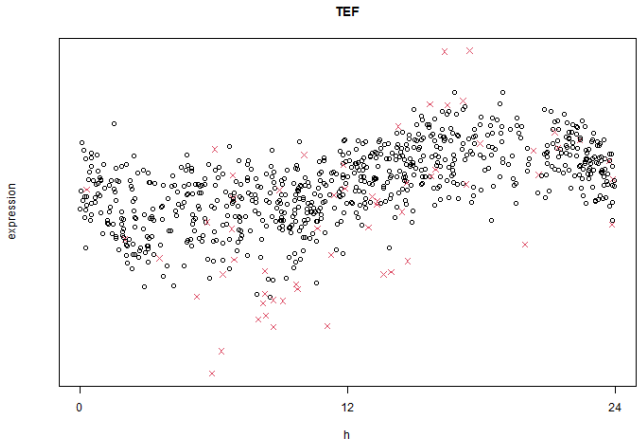
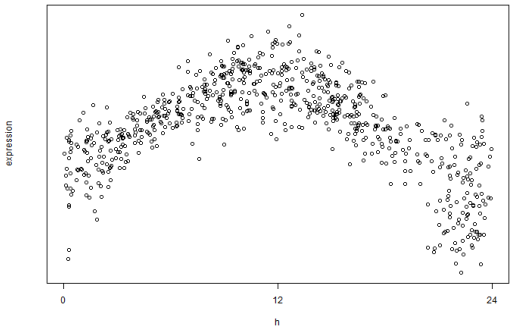
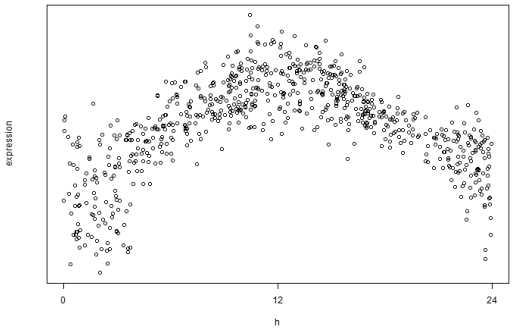


Figura 3.11: Datos de expresión ordenados para el gen *TEF* en músculo. En rojo se marcan las los *outliers*.

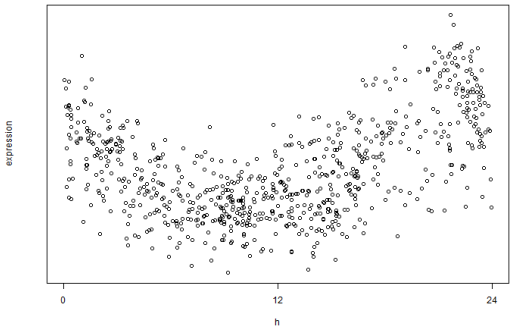
Una vez descartadas las muestras identificadas como *outliers*, los datos se normalizan y ordenan de nuevo a partir de los 12 genes *core* utilizando el procedimiento CPCA. Tras ello los datos se sincronizan según lo descrito en la sección 2.5.2. La figura 3.12 ilustra el efecto de la sincronización para los genes *ARNTL* y *DBP*.



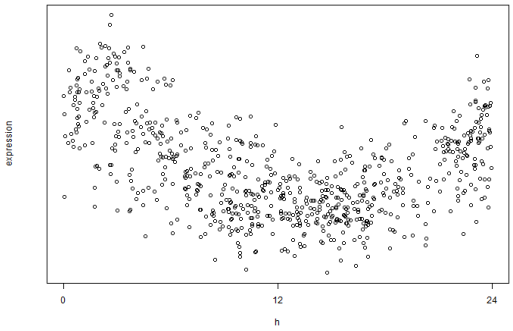
(a) *ARNTL* antes de la sincronización



(b) *ARNTL* después de la sincronización



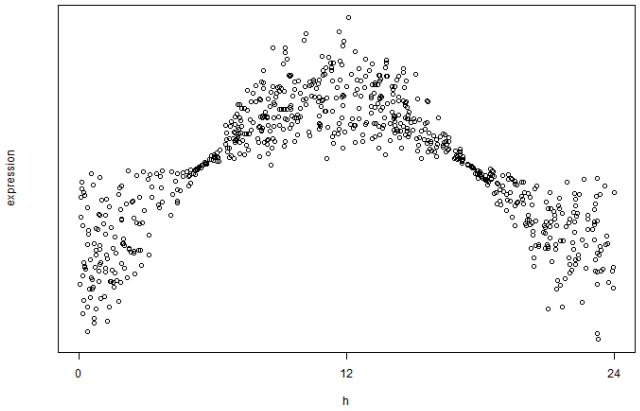
(c) *DBP* antes de la sincronización



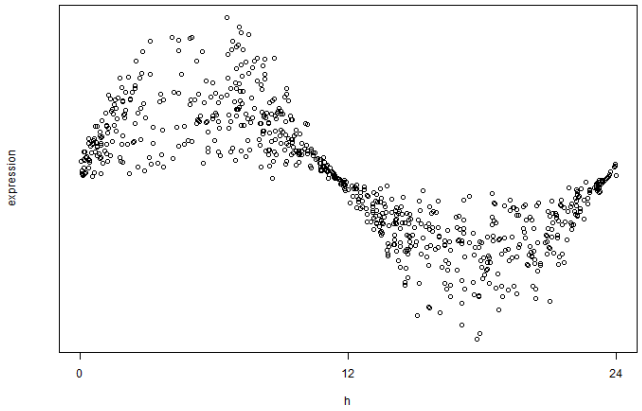
(d) *DBP* después de la sincronización

Figura 3.12: *DBP* y *ARNTL* antes y después de la sincronización.

La Figura 3.13 ilustra los patrones de los dos primeros *eigengenes* $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$. En ellos se observan los patrones que mayor variabilidad recogen para este tejido. Se puede observar que en ambos casos el punto de máxima expresión ocurre en el periodo luminoso del día.



(a) $\boldsymbol{E_1}$



(b) $\boldsymbol{E_2}$

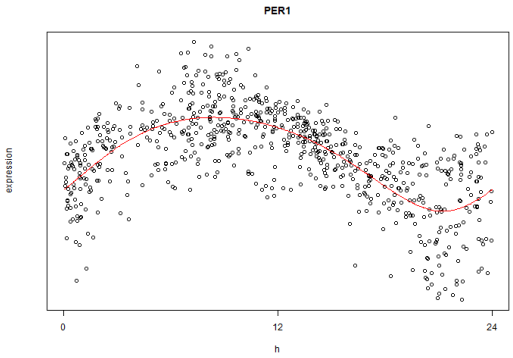
Figura 3.13: $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$ resultantes del método CPCA.

Las expresiones ordenadas para los 12 genes *core* se muestran en la figura 3.14. Se observa también

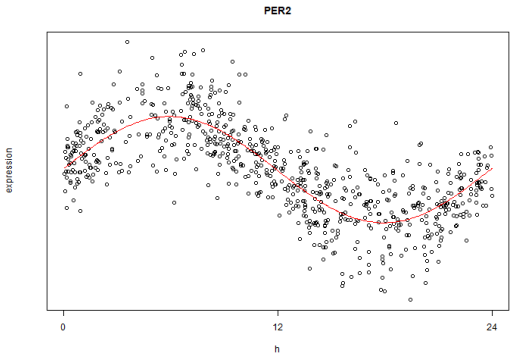
como los puntos de máxima expresión de *PER1, PER2, PER3, CRY1, CRY2* ocurren durante el periodo de luz [Korenčič et al., 2014] [Mure et al., 2018]. Concretamente, para este tejido, ninguno de los genes *core* tiene su momento de máxima expresión en horas de oscuridad. Por último, para estudiar la ritmicidad de los genes *core*, la Tabla 3.6 muestra los R^2 para el ajuste FMM de los 12 genes *core*. Se observa que como es de esperar, los genes que mejor R^2 tienen son los que presentan la misma forma que los *eigenenes* $\boldsymbol{E_1}$ y $\boldsymbol{E_2}$.

PER1	*PER2*	*PER3*	*CRY1*	*CRY2*	*ARNTL*
0.5128	0.6201	0.7373	0.5250	0.4846	0.6950
CLOCK	*NR1D1*	*RORA*	*DBP*	*TEF*	*STAT3*
0.5802	0.2004	0.6388	0.5233	0.5001	0.4434

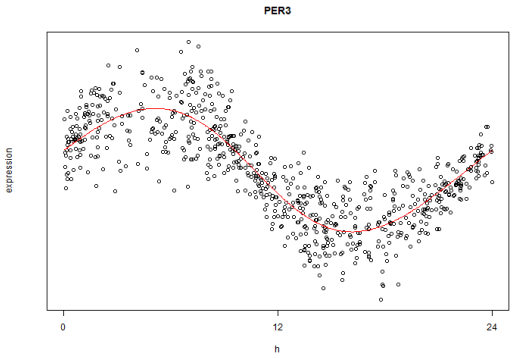
Tabla 3.6: R^2 para los 12 genes *core*.



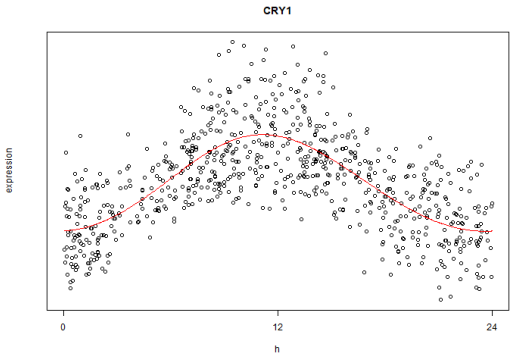
(a) *PER1*



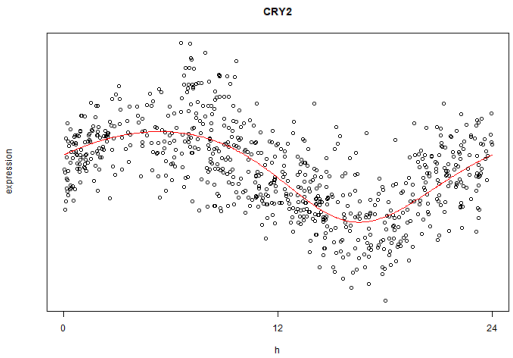
(b) *PER2*



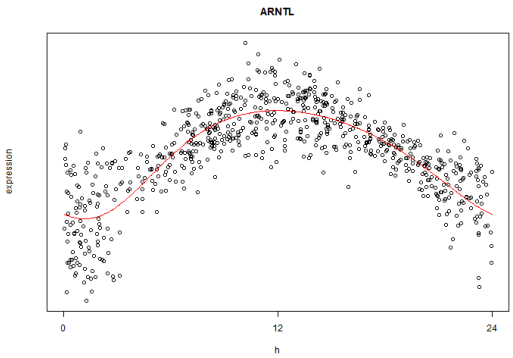
(c) *PER3*



(d) *CRY1*



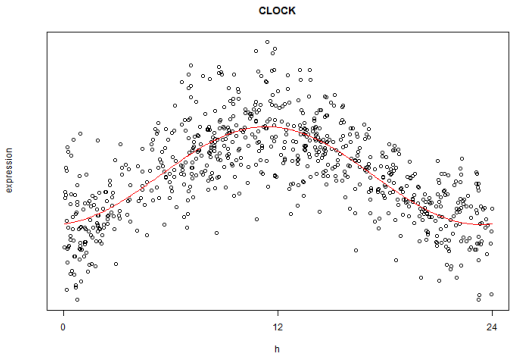
(e) *CRY2*



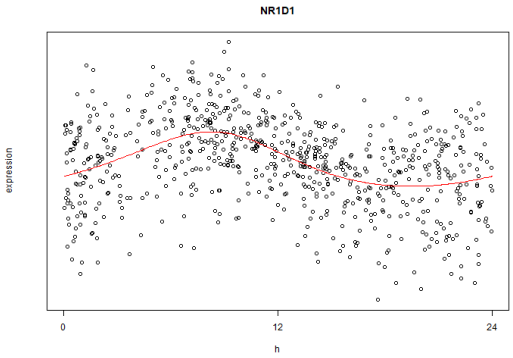
(f) *ARNTL*

Figura 3.14: Patrones de expresión de los genes *core* en tejido músculo esquelético.

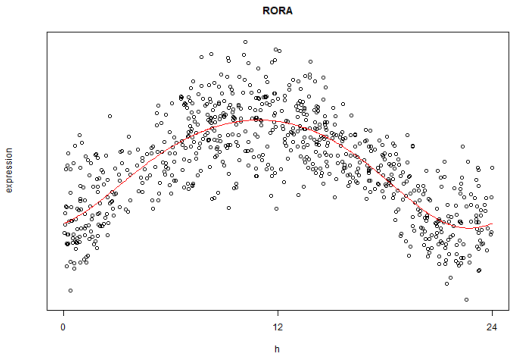
En rojo se muestra el ajuste de estos datos con el modelo FMM.



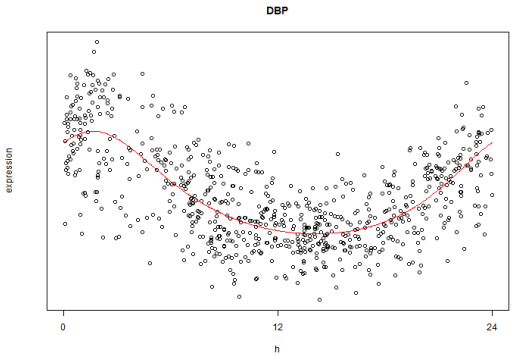
(g) *CLOCK*



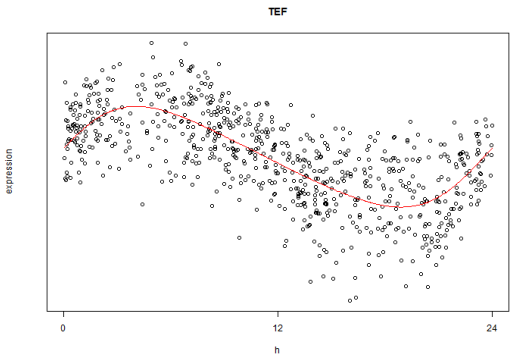
(h) *NR1D1*



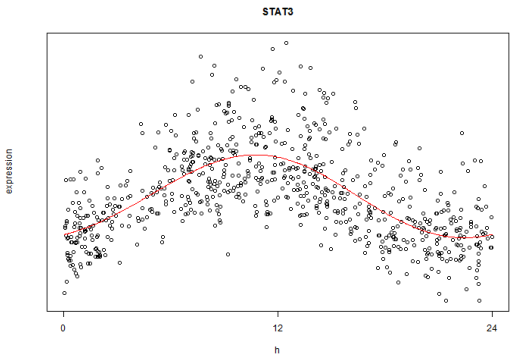
(i) *RORA*



(j) *DBP*



(k) *TEF*



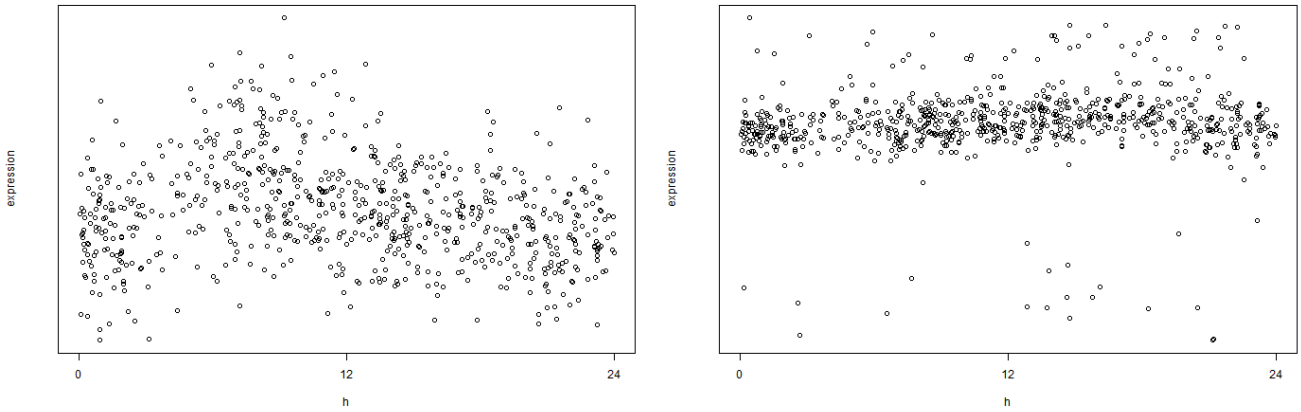
(l) *STAT3*

Figura 3.14: Patrones de expresión de los genes *core* en tejido músculo esquelético.

En rojo se muestra el ajuste de estos datos con el modelo FMM.

Obtención de los genes rítmicos

En este paso se ajusta el modelo FMM a los 15312 datos de expresión a lo largo 744 instantes de tiempo. En la Figura 3.14 se ven los ajustes FMM en los genes core. A continuación, se eliminan los genes que tengan $R^2 < 0.4$ o $\omega < 0.1$ por tratarse de genes potencialmente no rítmicos. En la Figura 3.15 se muestran dos genes que se eliminan al utilizar este criterio. El primer gen tiene demasiado ruido para ser considerado en las etapas siguientes y el segundo gen es plano por lo que los descartamos. Tras este paso, se tienen 2438 genes, una proporción de 0.1592 (IC $[0.1534, 0.1650]$) respecto a los 15312 genes.



(a) Gen eliminado por $R^2 < 0.4$

(b) Gen eliminado por $\omega < 0.1$

Figura 3.15: Dos genes descartados para cada criterio.

Enriquecimiento funcional

La clasificación no supervisada de los modelos ajustados permite identificar 6 *clusters* de genes con comportamiento similar. Cada *cluster* se describe por las estimaciones promedio de los genes que lo componen, ponderando con el correspondiente R^2 (Tabla 3.7). Gráficamente, estos patrones se muestran en la Figura 3.16. Algunos de estos cluster se puede identificar con uno de los 12 genes *core*.

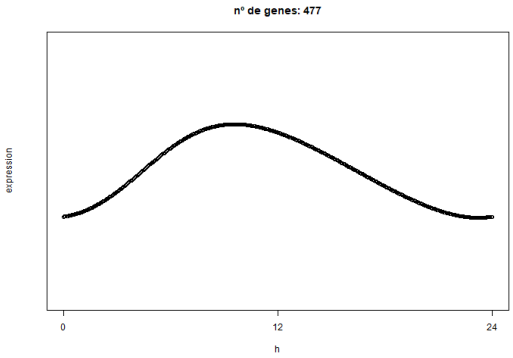
Los *clusters* 1 y 3 son prácticamente iguales salvo en el parámetro β: el 1 tiene un comportamiento *down-up* más marcado mientras que el 3 es ligeramente más simétrico. También son bastante diferentes respecto de ω, siendo algo más suave el patrón del *cluster* 1. El patrón 4 es todavía más simétrico y la principal diferencia se refiere a α, el punto de máxima expresión está algo más próximo a las 12 horas. El 2 es el que presenta mayores diferencias de los 4 primeros, con mayor amplitud, mayor anchura y el punto máximo muy cerca de las 12 horas. En cuanto a los momentos de máxima expresión se puede decir que estos cuatro primeros *clusters* contienen a los genes que se expresan en el periodo activo.

Se observa como el *cluster* 6 presenta un patrón similar al del gen *DBP*, es decir, engloba los genes que tienen su punto de máxima expresión después de los genes ROR (*ARNTL, NPAS, CLOCK*) [Wu et al., 2018, Mavroudis et al., 2018]. En el *cluster* 5 se agrupan los genes con el mismo patrón que los genes *ARNTL* y *CRY1*, es decir aquellos que tienen su punto de máxima expresión antes del comienzo del periodo inactivo.

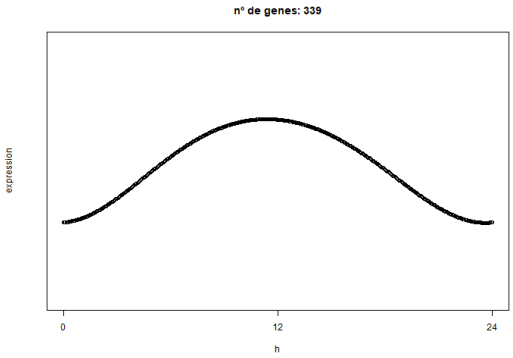
cluster	A	α	β	ω
1	0.3623 ± 0.0268	4.2955 ± 1.0785	1.5690 ± 1.0956	0.8017 ± 0.1261
2	0.4020 ± 0.0361	3.2336 ± 2.0070	0.2168 ± 2.0534	0.8556 ± 0.1216
3	0.3094 ± 0.0334	4.6941 ± 0.7605	2.2273 ± 0.8157	0.7487 ± 0.1194
4	0.3058 ± 0.0289	5.6700 ± 1.9044	2.8563 ± 1.9169	0.8660 ± 0.1263
5	0.3521 ± 0.0412	1.5076 ± 1.3186	4.6072 ± 1.3255	0.8392 ± 0.1128
6	0.4101 ± 0.0592	2.1679 ± 1.0230	1.9010 ± 1.1385	0.7496 ± 0.1103

Tabla 3.7: Valores (promedio $\pm$ desviación típica) de los parámetros FMM de cada *cluster*.

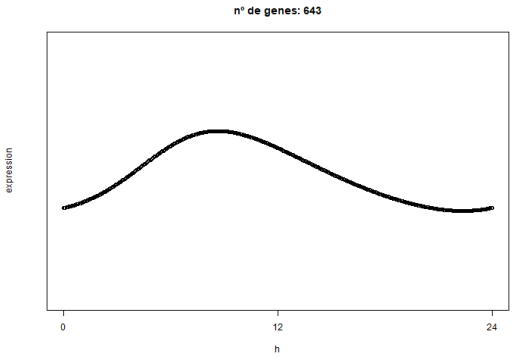
Los valores se calculan como medias y desviaciones típicas ponderadas, utilizando el R^2 como peso para cada gen.



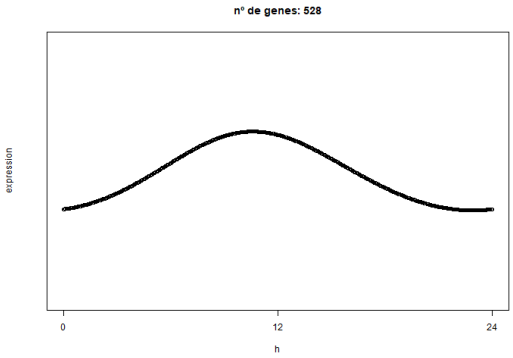
(a) *Cluster* 1: 477 genes



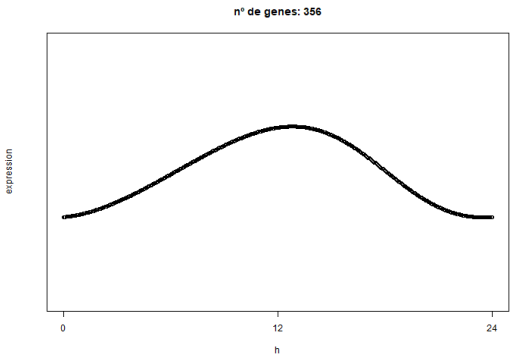
(b) *Cluster* 2: 339 genes



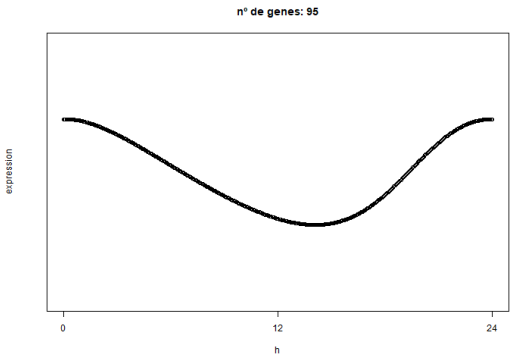
(c) *Cluster* 3: 643 genes



(d) *Cluster* 4: 528 genes



(e) *Cluster* 5: 356 genes



(f) *Cluster* 6: 95 genes

Figura 3.16: Patrón rítmico promedio para cada *cluster*.

La ontología GO está formada a su vez por 3 ontologías: **función molecular** (MF), **componente celular** (CC) y **proceso biológico** (BP). En este trabajo se ha restringido el enriquecimiento funcional a la ontología BP, puesto que, presumiblemente, es la más sencilla de interpretar para alguien no experto en biología molecular. El principal objetivo será tratar de establecer una posible relación biológica entre los genes de cada *cluster* y el comportamiento rítmico en sus niveles de expresión. Todas las discusiones presentadas a continuación se han hecho sin consultar a un experto en la materia, por lo que el principal enfoque es presentar el análisis e ilustrar su utilidad. Con la colaboración de un experto en el tema se podrían sacar conclusiones mucho más elaboradas y ricas.

***Cluster* 1** :

En este *cluster* (Figura 3.16a) se observa como los genes rítmicos están implicados mayoritariamente en procesos de regulación (Figura 3.17). En la regulación de la expresión génica se incluyen remodeladores de la cromatina, modificación de histonas, factores de transcripción, etc. La mayor parte activan la transcripción. También genes implicados en regulación de procesos metabólicos, en la síntesis de ácidos nucleicos, etc. Unos pocos genes parecen estar implicados en la inhibición de la síntesis de proteínas. Se podría interpretar, por tanto, que los genes pertenecientes a este cluster están implicados en la regulación de procesos biológicos fundamentales para la célula que podría esperarse tengan una expresión rítmica.

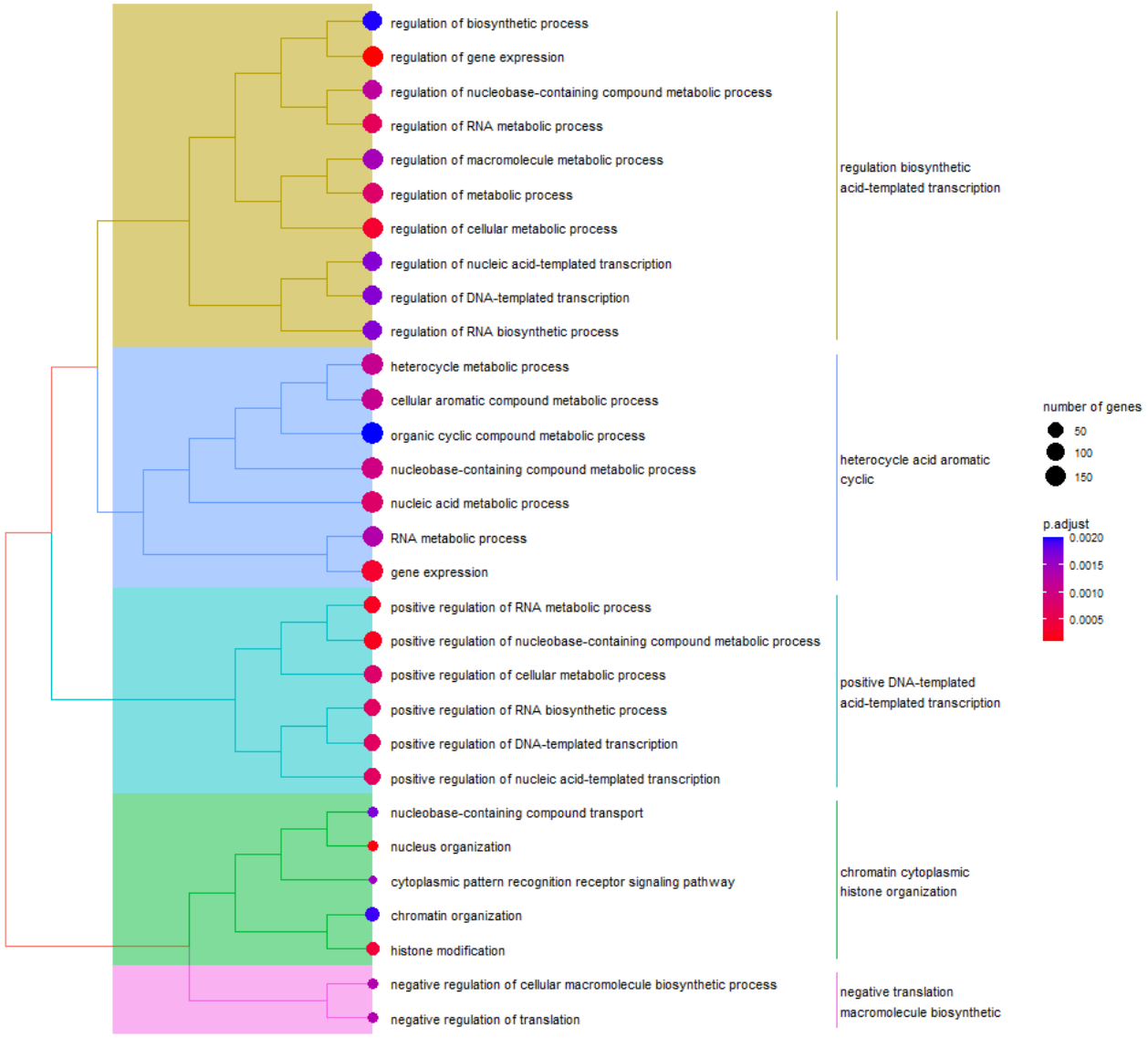


Figura 3.17: Resultados del enriquecimiento funcional utilizando la ontología BP de la base de datos GO para el *cluster* 1.

Cluster 2 :

En este *cluster* (Figura 3.16b) la categoría más numerosa está implicada de nuevo en regulación de diversos procesos (Figura 3.18): Regulación de procesos celulares, del metabolismo, de procesos de síntesis, etc. Otros genes parecen estar implicados en activación del ciclo celular, en procesos de síntesis de lípidos, en modificaciones y degradación de proteínas, en el metabolismo y reparación de DNA, etc. De nuevo, es de esperar que genes reguladores relacionados con el ciclo celular y el metabolismo tengan ritmicidad.

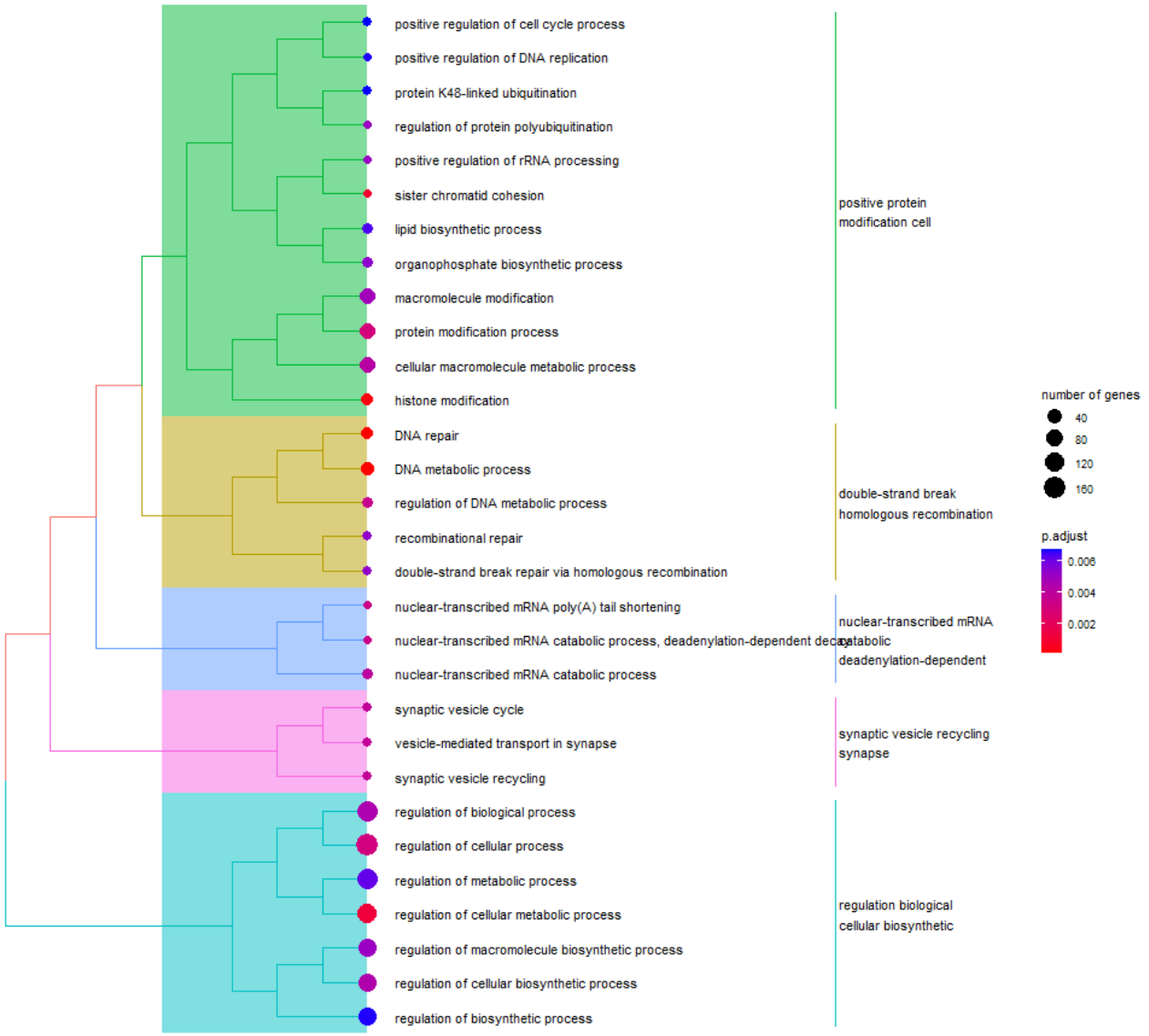


Figura 3.18: Resultados del enriquecimiento funcional utilizando la ontología BP de la base de datos GO para el *cluster* 2.

***Cluster* 3** :

En este *cluster* (Figura 3.16c) los genes rítmicos parecen estar implicados en el metabolismo de ácidos nucleicos, en el procesamiento del RNA y en su regulación, en la modificación de histonas y por tanto en la remodelación de la cromatina, etc (Figura 3.19) . Todos estos procesos celulares tienen que ver en último término con la regulación de la expresión génica que debe estar estrechamente controlada, quizás por patrones rítmicos.

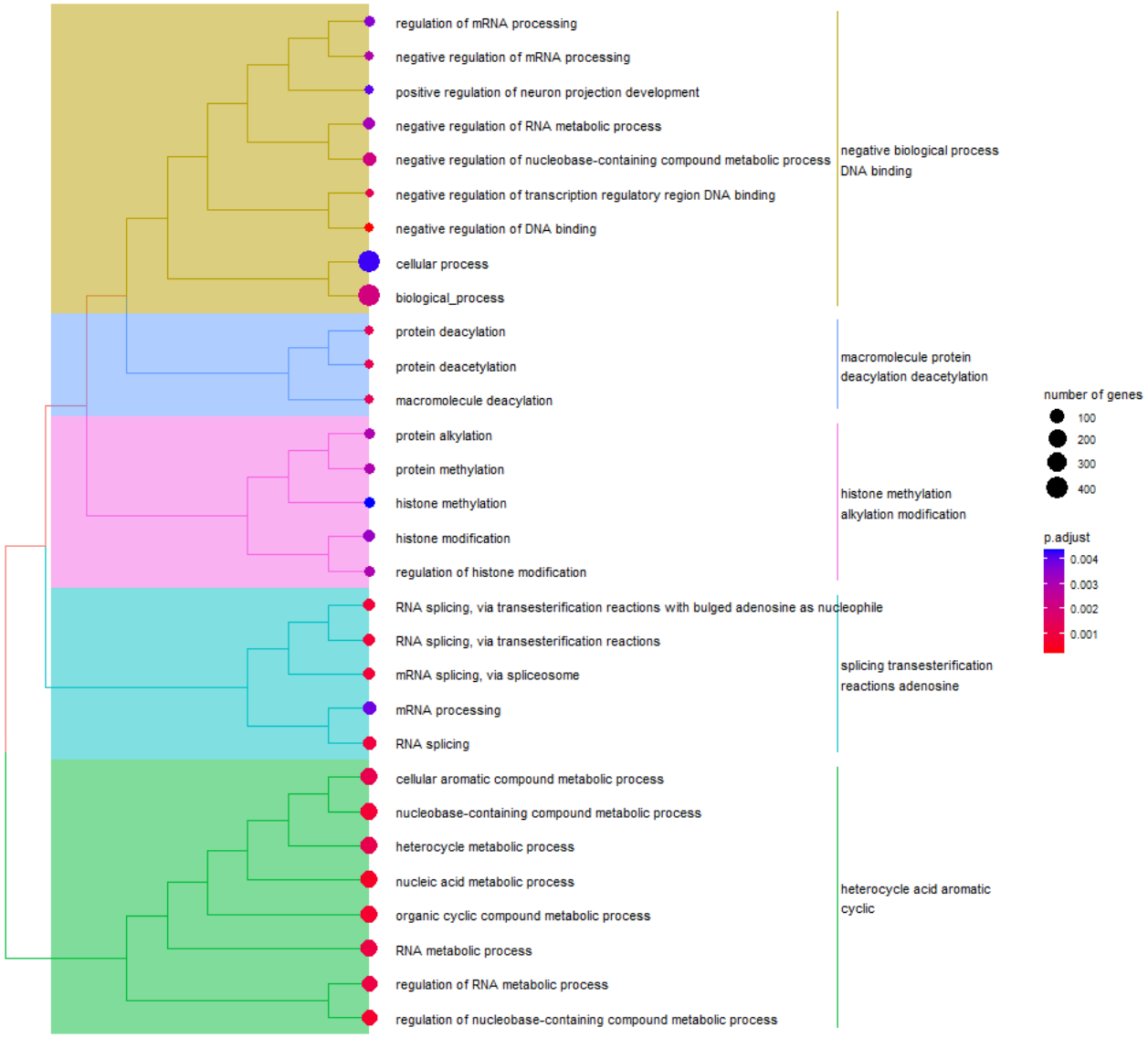


Figura 3.19: Resultados del enriquecimiento funcional utilizando la ontología BP de la base de datos GO para el *cluster* 3.

***Cluster* 4** :

En este *cluster* (Figura 3.16d) se encuentran genes que parecen estar implicados en la regulación de procesos de fosforilación inhibiendo proteínas quinasas (Figura 3.20). La fosforilación es una de las principales modificaciones que sufren las proteínas, por ejemplo las enzimas, para regular su actividad. También aparecen genes implicados en transporte de iones de potasio y en su regulación y por tanto en el mantenimiento del potencial de membrana. Otros de estos genes rítmicos intervienen en la regulación de la motilidad celular. Sería interesante estudiar por qué genes implicados en procesos que ocurren en la membrana celular son rítmicos.

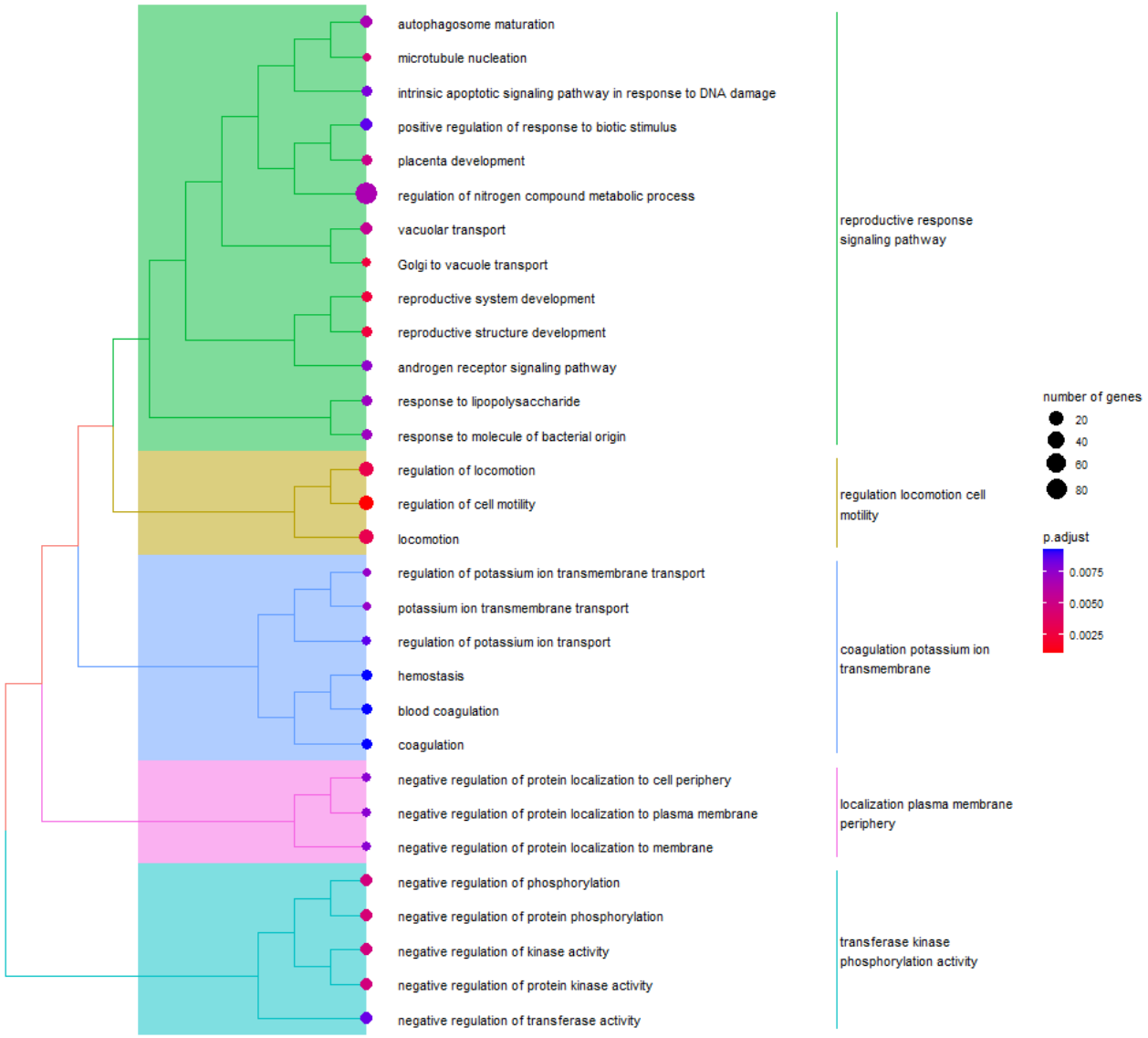


Figura 3.20: Resultados del enriquecimiento funcional utilizando la ontología BP de la base de datos GO para el *cluster* 4.

***Cluster* 5** :

En este *cluster* (Figura 3.16e) los genes rítmicos participan en el almacenamiento y metabolismo de lípidos, síntesis de lípidos de membrana, regulación de canales de potasio, regulación de la expresión génica con remodelación de la cromatina y modificación de histonas, etc (Figura 3.21). Estos resultados parecen mostrar que la mayoría de los genes que intervienen en funciones esenciales para la célula muestran expresión rítmica.

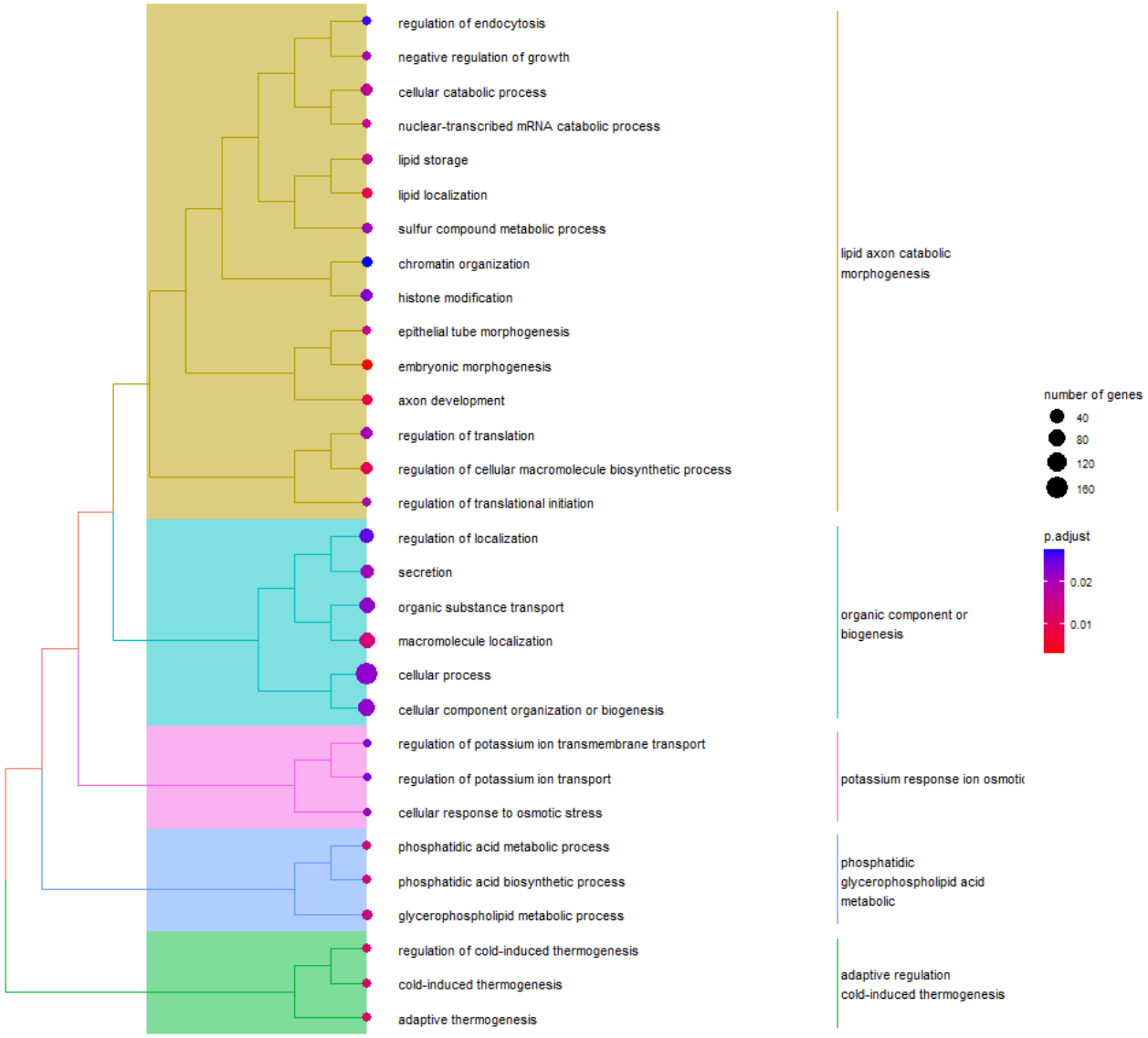


Figura 3.21: Resultados del enriquecimiento funcional utilizando la ontología BP de la base de datos GO para el *cluster* 5.

***Cluster* 6** :

Por último, en este *cluster* (Figura 3.16f), aparecen sobre-representados genes implicados en la respuesta al daño (Figura 3.22), que se define en GO como un proceso que cambia el estado o actividad de la célula como resultado de un estímulo indicando un daño al organismo. Sería de interés estudiar por qué algunos genes que regulan estos procesos son rítmicos en el tejido muscular. Además en este *cluster* aparecen genes implicados en migración, división mitótica y ciclo celular. Todos estos procesos son cíclicos y están muy regulados.

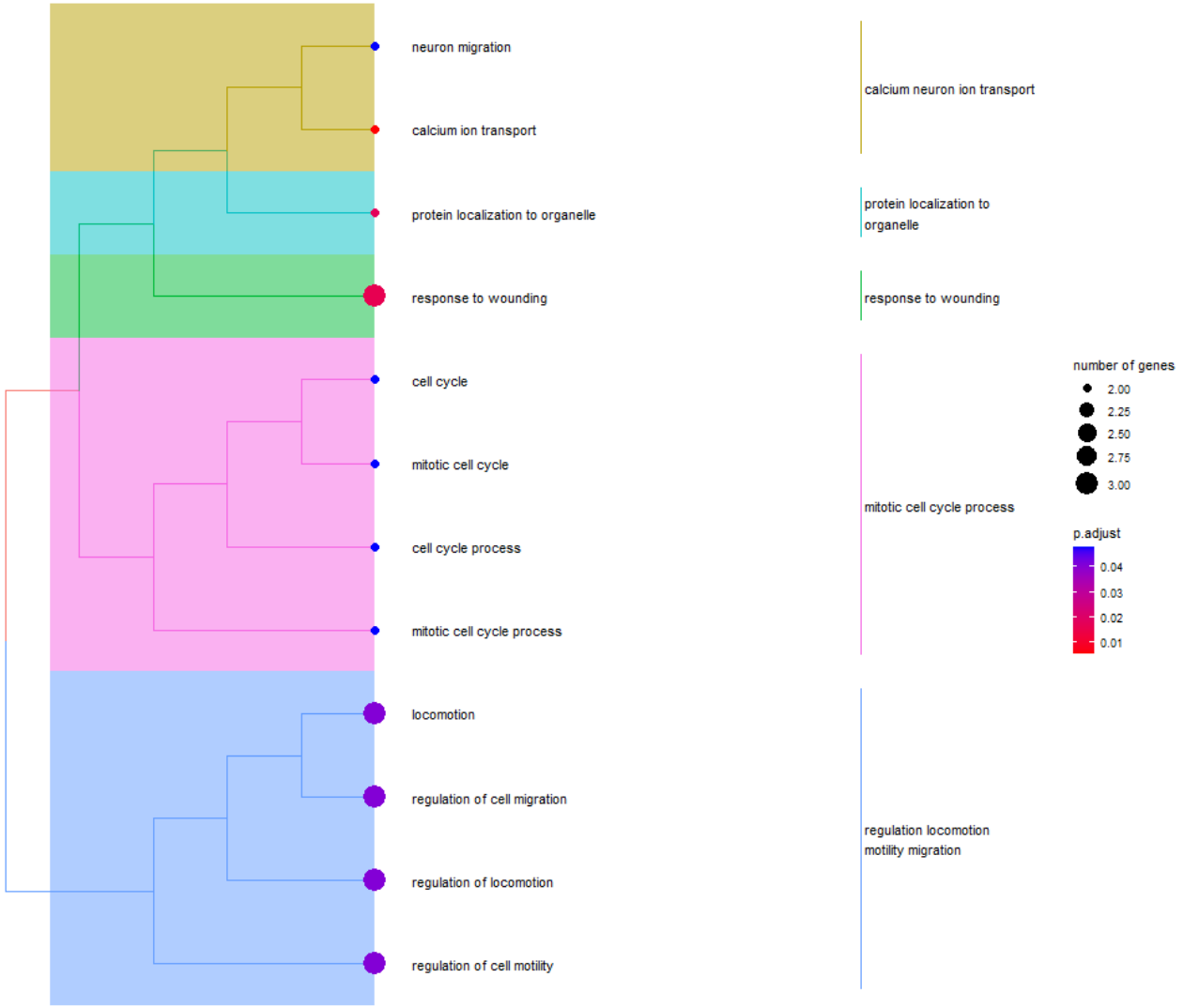


Figura 3.22: Resultados del enriquecimiento funcional utilizando la ontología BP de la base de datos GO para el *cluster* 6.

Discusión de resultados y conclusiones

En este trabajo se propone un metodología para el análisis de la ritmicidad en datos de expresión génica. Este procedimiento ha permitido estudiar los genes rítmicos y clasificarlos con la finalidad de estudiar su función biológica en el tejido humano músculo esquelético, a partir de datos de expresión post-mortem de la base de datos GTEX [Lonsdale et al., 2013].

Trabajar con datos post-mortem supone tener que resolver el problema de la estimación del orden temporal. Para estos datos no se conoce de manera precisa el TOD. En este trabajo se han propuesto tres métodos para obtener la estimación del orden temporal y se ha comparado su funcionamiento. Mediante un estudio de simulación, se ha comprobado que el método CPCA, a través del cálculo de los *eigengenes*, puede ser el más indicado, puesto que es el de menor coste computacional y permite trabajar con datos no equiespaciados, consiguiendo resultados muy similares al resto.

El algoritmo que se ha propuesto en este trabajo para abordar el análisis de expresiones génicas consta de cuatro etapas. En la primera etapa se proponen unos pasos para eliminar aquellos genes cuya expresión aporta poca información y preparar los datos para que tengan mayor interpretabilidad en las siguientes etapas. En la segunda etapa, se estima el orden de las muestras a partir de un conjunto de genes típicamente rítmicos, denominados *cores*. Se trata de un orden común para todos los genes que incorpora la decisión de eliminar aquellas muestras con excesiva influencia en dicho orden. Una vez establecido el orden, se analiza la ritmicidad de los genes basada en el modelo FMM. El estudio de los parámetros de este modelo permite identificar genes con un marcado comportamiento rítmico biológicamente relevantes. Finalmente, en un intento de comprender profundamente esta información los genes se agrupan atendiendo a la similitud de sus patrones rítmicos. A estos grupos se les da interpretación biológica mediante un análisis de enriquecimiento funcional.

Se ha aplicado el algoritmo al tejido músculo esquelético y se ha visto que se seleccionan 2438 genes rítmicos de 56200 genes iniciales. Los puntos de máxima expresión estimados en los genes *core* coinciden con el conocimiento biológico de estos genes. Analizando esta bondad con una medida como el R^2 se obtiene que para este conjunto de genes esa medida la mayoría tiene un $R^2 > 0.5$.

La utilización de un algoritmo de *clustering* aplicado a datos funcionales ha permitido agrupar los genes identificados como rítmicos en 6 patrones diferenciados para el tejido músculo esquelético.

Se ha mostrado cómo el enriquecimiento funcional puede aportar información biológica acerca de qué categorías están sobre-representadas en cada cluster y esto se puede relacionar con el tipo de gen que hay en cada uno de ellos con la ayuda de un experto en la materia. Se ha observado que la mayoría de categorías sobre-representadas tienen relación con procesos celulares que en principio, podrían suponerse cíclicos, lo que de alguna forma, puede utilizarse para validar el procedimiento de identificación de la ritmicidad.

4.1. Trabajo futuro

Las dos principales líneas futuras de investigación son las siguientes:

- Evaluación del efecto de variables fenotípicas tales como edad y sexo: Una de las cuestiones interesantes que quedan por resolver en este trabajo es el efecto de estas variables en los comportamientos cíclicos. Muchos autores han tratado de estableces diferencias en estos comportamientos en tejidos y condiciones muy específicas, que a priori se espera que sean muy dependientes de estas variables, sin haber obtenido resultados concluyentes. En este trabajo, se ha realizado una aproximación preliminar en la que se han estudiado los residuos del modelo FMM ajustado, como un primer indicio del interés de incluir estas variables en este procedimiento, sin obtener resultados satisfactorios. Una manera de abordarlo sería incluir variables explicativas al modelo FMM.
- Como se ha visto en los resultados del enriquecimiento funcional, este aporta una gran cantidad de información sobre las funcionalidades que describen los genes. En el futuro sería de interés continuar la investigación a partir de estos resultados, con ayuda de un experto, para correctamente identificar la importancia de los resultados y para recopilar las categorías relevantes para cada grupo de genes en función de la forma del patrón medio del grupo.
- Aplicar el procedimiento a diferentes tejidos para evaluar las diferencias, así como tratar de generalizar los hallazgos a otras especies y bases de datos.
- Todos los desarrollos de software se podrían implementar en un paquete abierto al uso.

Bibliografía

[Alter et al., 2000] Alter, O., Brown, P. O., and Botstein, D. (2000). Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences*, 97(18):10101–10106.

[Anafi et al., 2017] Anafi, R. C., Francey, L. J., Hogenesch, J. B., and Kim, J. (2017). Cyclops reveals human transcriptional rhythms in health and disease. *Proceedings of the National Academy of Sciences*, 114(20):5312–5317.

[Ashburner et al., 2000] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., et al. (2000). Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29.

[Central et al., 2023] Central, G., Aleksander, S. A., Balhoff, J., Carbon, S., Cherry, J. M., Drabkin, H. J., Ebert, D., Feuermann, M., Gaudet, P., Harris, N. L., et al. (2023). The gene ontology knowledgebase in 2023. *Genetics*, 224(1).

[Cornelissen, 2014] Cornelissen, G. (2014). Cosinor-based rhythmometry. *Theoretical Biology and Medical Modelling*, 11(1):1–24.

[Curtis and FitzGerald, 2006] Curtis, A. M. and FitzGerald, G. A. (2006). Central and peripheral clocks in cardiovascular and metabolic function. *Annals of medicine*, 38(8):552–559.

[He et al., 2022] He, S., Ye, H., and He, K. (2022). Spline estimation of functional principal components via manifold conjugate gradient algorithm. *Statistics and Computing*, 32(6):106.

[Hollander et al., 1999] Hollander, M., Wolfe, D. A., and Chicken, E. (1999). *Nonparametric statistical methods*, volume 1. Wilfey, New York.

[Jammalamadaka and SenGupta, 2001] Jammalamadaka, S. R. and SenGupta, A. (2001). *Topics in circular statistics*, volume 5. world scientific.

[Kirby and Miranda, 1996] Kirby, M. J. and Miranda, R. (1996). Circular nodes in neural networks. *Neural Computation*, 8(2):390–402.

[Korenčič et al., 2014] Korenčič, A., Košir, R., Bordyugov, G., Lehmann, R., Rozman, D., and Herzel, H. (2014). Timing of circadian genes in mammalian tissues. *Scientific reports*, 4(1):1–9.

[Larriba et al., 2022] Larriba, Y., Mason, I. C., Saxena, R., Scheer, F. A., and Rueda, C. (2022). Circust: a novel methodology for temporal order reconstruction of molecular rhythms; validation and application towards a daily rhythm gene expression atlas in humans. *bioRxiv*, pages 2022–12.

[Larriba et al., 2020] Larriba, Y., Rueda, C., Fernández, M. A., and Peddada, S. D. (2020). Order restricted inference in chronobiology. *Statistics in Medicine*, 39(3):265–278.

[Leng et al., 2015] Leng, N., Chu, L.-F., Barry, C., Li, Y., Choi, J., Li, X., Jiang, P., Stewart, R. M., Thomson, J. A., and Kendziorski, C. (2015). Oscope identifies oscillatory genes in unsynchronized single-cell rna-seq experiments. *Nature methods*, 12(10):947–950.

[Levi and Schibler, 2007] Levi, F. and Schibler, U. (2007). Circadian rhythms: mechanisms and therapeutic implications. *Annual Review of Pharmacology and Toxicology*, 47:593–628.

[Liu et al., 2017] Liu, Z., Lou, H., Xie, K., Wang, H., Chen, N., Aparicio, O. M., Zhang, M. Q., Jiang, R., and Chen, T. (2017). Reconstructing cell cycle pseudo time-series via single-cell transcriptome data. *Nature communications*, 8(1):22.

[Lonsdale et al., 2013] Lonsdale, J., Thomas, J., Salvatore, M., Phillips, R., Lo, E., Shad, S., Hasz, R., Walters, G., Garcia, F., Young, N., et al. (2013). The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580–585.

[Mavroudis et al., 2018] Mavroudis, P. D., DuBois, D. C., Almon, R. R., and Jusko, W. J. (2018). Modeling circadian variability of core-clock and clock-controlled genes in four tissues of the rat. *PLoS One*, 13(6):e0197534.

[Mure et al., 2018] Mure, L. S., Le, H. D., Benegiamo, G., Chang, M. W., Rios, L., Jillani, N., Ngotho, M., Kariuki, T., Dkhissi-Benyahya, O., Cooper, H. M., et al. (2018). Diurnal transcriptome atlas of a primate across major neural and peripheral tissues. *Science*, 359(6381):eaao0318.

[Park and Jun, 2009] Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341.

[Pearson, 1901] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

[Press Release, 2017] Press Release (2017). Nobel prize outreach 2017. `https://www.nobelprize.org/prizes/medicine/2017/press-release/`. visitado 19-06-2023.

[Reynolds et al., 2006] Reynolds, A. P., Richards, G., de la Iglesia, B., and Rayward-Smith, V. J. (2006). Clustering rules: a comparison of partitioning and hierarchical clustering algorithms. *Journal of Mathematical Modelling and Algorithms*, 5:475–504.

[Ruben et al., 2018] Ruben, M. D., Wu, G., Smith, D. F., Schmidt, R. E., Francey, L. J., Lee, Y. Y., Anafi, R. C., and Hogenesch, J. B. (2018). A database of tissue-specific rhythmically expressed human genes has potential applications in circadian medicine. *Science translational medicine*, 10(458):eaat8806.

[Rueda et al., 2019] Rueda, C., Larriba, Y., and Peddada, S. D. (2019). Frequency modulated möbius model accurately predicts rhythmic signals in biological and physical sciences. *Scientific Reports*, 9(1):18701.

[Scholz, 2007] Scholz, M. (2007). Analysing periodic phenomena by circular pca. In *Bioinformatics Research and Development: First International Conference, BIRD 2007, Berlin, Germany, March 12-14, 2007. Proceedings*, pages 38–47. Springer.

[Schubert and Rousseeuw, 2019] Schubert, E. and Rousseeuw, P. J. (2019). Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In *Similarity Search and Applications: 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2–4, 2019, Proceedings 12*, pages 171–187. Springer.

[Subramanian et al., 2005] Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., et al. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550.

[Wu et al., 2018] Wu, G., Ruben, M. D., Schmidt, R. E., Francey, L. J., Smith, D. F., Anafi, R. C., Hughey, J. J., Tasseff, R., Sherrill, J. D., Oblong, J. E., et al. (2018). Population-level rhythms in human skin with implications for circadian medicine. *Proceedings of the National Academy of Sciences*, 115(48):12313–12318.

[Zhang et al., 2014] Zhang, R., Lahens, N. F., Ballance, H. I., Hughes, M. E., and Hogenesch, J. B. (2014). A circadian gene expression atlas in mammals: implications for biology and medicine. *Proceedings of the National Academy of Sciences*, 111(45):16219–16224.

Anexo 1: Metodología adicional

5.1. Media recortada

Se define la media recortada para $\boldsymbol{X_i}$

$$\bar{X}_i = \frac{1}{\lfloor m - m \times r \rfloor} \sum_{j=1}^{\lfloor m - m \times r \rfloor} x_{i,(j)} \tag{5.1}$$

Donde $x_{i,(j)}$ es la observación j-ésima del vector de expresiones ordenadas de menor a mayor del gen i y r es un valor entre 0 y 1 indicando la proporción de elementos a recortar al calcular la media.

5.2. Residuos estandarizados

Se definen los residuos estandarizados para una gen i de la matriz $[X]$ como:

$$rstd_{i,j} = \frac{r_{i,j}}{\sqrt{\frac{1}{m}\sum_{j=1}^{m}(x_{i,j} - \hat{x_{i,j}})^2}} \tag{5.2}$$

Donde $r_{i,j}$ representa el residuo del la observación j en el gen i y $\hat{x_{i,j}}$ son los valores predichos.

5.3. Clustering

Los algoritmos de clustering son algoritmos no supervisados para la clasificación de elementos en función de su similaridad. En este proyecto se utiliza un algoritmo de clustering basado en centroides.

5.3.1. K-meadoids

K-meadoids es una versión robusta del algoritmo de clustering K-means . Se define la matriz $[X]_{n \times p}$, de n elementos con p variables y se escoge un $k =$ número de *clusters* $(k < n)$. El algoritmo utiliza una medida de disimilaridad para realizar el clustering. Una medida válida es la distancia euclidea que se define para un elemento i y un elemento j. [Park and Jun, 2009]

$$d_{i,j} = \sqrt{\sum_{a=1}^{p}(X_{i,a} - X_{j,a})^2}; i = 1, ..., n; j = 1, ..., n \tag{5.3}$$

Una de las diferencias principales entre k-means y k-medoids es que en k-medoids los medoids o puntos centrales del *cluster* son elementos de los datos mientras que en k-means estos son el centro geométrico del *cluster*. Esto además hace los resultados más interpretables, dando como centro un valor existente que se puede usar como referencia para el *cluster*. Para escoger los medoids iniciales se utiliza la siguiente medida:

$$\upsilon_j = \sum_{i=1}^{n} \frac{d_{i,j}}{\sum_{l=1}^{n} d_{i,l}}; j = 1, ..., n \tag{5.4}$$

Se escogen como centros los K elementos con menor valor de υ. A cada elemento se le asigna al medoid que tenga más cercano. También se puede escoger k elementos aleatorios como medoids iniciales.

Existen dos algoritmos principales para realizar el clustering [Schubert and Rousseeuw, 2019]:

- Partitioning Arround Medoids (PAM): Se define una función objetivo para cada *cluster*, por ejemplo la suma de distancias de elementos al centro del *cluster*. Para cada elemento del *cluster* se computa esta función objetivo cuando este pasa a ser el nuevo medoid. Se escoge como nuevo medoid del *cluster* el elemento que minimiza la función objetivo. Se vuelven a formar los nuevos *clusters* asignando cada elemento al medoid más cercano. Se repite el paso hasta que no cambien ningún medoid.

- Clustering Large Applications (CLARA): se aplica el algoritmo PAM repetidamente sobre un subconjunto de tamaño n'. El valor recomendado es $n' = 40 + 2k$. Para cada uno de estos subconjuntos se calcula la función objetivo global, por ejemplo la suma de las funciones objetivo de cada *cluster*, con todos los datos y se devuelve los medoids que la minimizan.

Para datasets de más de 2000 observaciones se recomienda usar CLARA, ya que al solo realizar el algoritmo PAM sobre un subconjunto este es más rápido [Reynolds et al., 2006].

Para escoger el número de *clusters* óptimos se ajusta el problema para cada $k \in (k_1, ..., k_n)$ y se escoge el k que minimiza la media de la silueta para todas las observaciones. Definimos la silueta de un elemento i como:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i)}, i = 1, ..., n \tag{5.5}$$

Donde $a(i)$ es la media de la disimilaridades para i con el resto de elementos de su *cluster* y $b(i)$ es el mínimo de las media de disimilaridades entre el elemento i y los elementos de un *cluster* al que no pertenece para todos los *clusters*s. Por tanto $b(i)$ representa la dismilaridad media de i con su *cluster* ”vecino”, es decir el *cluster* más parecido a i.

Anexo 2: Código desarrollado

En este anexo se incluye el código R y Matlab que se ha desarrollado en este trabajo.En primer lugar se incluye un fichero de codigo R con funciones que se usan en todo el trabajo, segundo los RMarkdown para la simulación y etapas del algoritmo y por último el codigo Matlab de la red neuronal. Para el código Matlab se necesita descargar los archivos donde se implementa la red que se proporcionan en: `https://github.com/matthias-scholz/nonlinear-pca`.

funcionesTFGEST.R

```
# funcion que elimina los genes que tengan mas de una proporcion de ceros
remove_zeroes <- function(gene_data, threshold = 0.40) {
  zeroCount <-
    apply(gene_data, 1, function(x)
      sum(x == 0)) / dim(gene_data)[2]
  validGenes <- attr(zeroCount, "names")[zeroCount < threshold]
}
#funcion que normaliza un vector
normalize <- function(x) {
  return((2 * (x - min(x)) / (max(x) - min(x))) - 1)
}
#funcion que estandariza un vector
standardize <- function(x) {
  return((x - mean(x) / sd(x)))
}
#funcion que centra un vector mediante su media
centre_mean <- function(x) {
  return(x - mean(x))
}
#funcion que calcula la media recortada
upper_trim_mean <- function(x, trim) {
  x <- sort(x)
  mean(x[1:floor(length(x) * (1 - trim))])
}
#funcion que la proporcion de genes de data que estan en core
score_core2 <- function(data, core) {
  index <- c(NULL)
  score = 0
```

```
  for (i in core) {
    index <- which(rownames(data) == i)
    if (length(index) != 0) {
      score <- score + 1
    }

  }
  return(score / length(core))
}
#funcion para simular los efectos de threshold en recortar los genes con
# un valor d
test_threshold <- function(threshold, d, core, gene_data) {
  s <- d > threshold
  return(c(mean(s), score_core2(gene_data[s, ], core)))
}
#funcion que devuelve los outliers tras ajustar un modelo FMM
FMMOutliers <- function(data) {
  model <- fitFMM(data)
  stdresid <- abs(resid(model)) / sqrt(modelSSE / length(data))
  return(stdresid < 3)
}
#funcion para 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))
}
#funcion que normaliza un angulo entre 0 y 2pi
normalize_angle <- function(angle) {
  return(angle - 2 * pi * floor(angle / (2 * pi)))
}

#funcion para obtener los valores predichos de un modelo FMM en la
   simulacion
get_fit_sim <- function(gene) {
  fit <- fitFMM(gene)
  M <- fitM
  A <- fitA
  alpha <- fitalpha
  beta <- fitbeta
  omega <- fitomega
  generateFMM(M,
              A,
              alpha,
```

```
            beta,
            omega,
            length.out = 500,
            plot = FALSE)
}
#funcion que calcula el and logico de un vector
and_logical <- function(data) {
  if (sum(data) == length(data))
    return(TRUE)
  else
    return(FALSE)
}
#funcion que calcula el or logico de un vector
or_logical <- function(data) {
  if (sum(data) >= 1)
    return(TRUE)
  else
    return(FALSE)
}
#funcion para calcular la desviacion tipica circular con pesos
weighted_sd_circ <- function(angle, w) {
  sum_w <- sum(w)
  x <- cos(angle)
  y <- sin(angle)
  sum_x <- sum(x * w)
  sum_y <- sum(y * w)
  sqrt(-2 * log(sqrt(sum_x ^ 2 + sum_y ^ 2) / sum_w))

}
```

Simulación

Samuel Bassols

2023-05-25

```
library(FMM)
library(FactoMineR)
setwd("D:/Universidad/TFG-Estadistica")
source("D:/Universidad/TFG-Estadistica/FuncionesTFGEST.R")
```

1.Simulación de los datos

```
set.seed(2023)
noise<-sqrt(0.05)
size<-500
sim1.1<-generateFMM(M = 0,A = 1,alpha = 0,beta = 3*pi/2,
                    omega = 0.9, plot=TRUE,
                    length.out = size,sigmaNoise = noise)
sim2.1<-generateFMM(M = 0,A = 1,alpha = 0,beta = 3*pi/2,
                    omega = 0.1, plot=TRUE,
                    length.out = size,sigmaNoise = noise)
sim3.1<-generateFMM(M = 0,A = 1,alpha = pi/2,beta = 3*pi/2,
                    omega = 0.9, plot=TRUE,
                    length.out = size,sigmaNoise = noise)
sim4.1<-generateFMM(M = 0,A = 1,alpha = pi,beta = 3*pi/2,
                    omega = 0.1, plot=TRUE,
                    length.out = size,sigmaNoise = noise)

sim5.1<-split(rnorm(size*6,0,noise),f=rep(1:6,500))
plot(sim5.1$"1")
data.1<-t(data.frame(sim1.1$y,sim2.1$y,sim3.1$y,sim4.1$y,
                     sim5.1$"1",sim5.1$"2",sim5.1$"3",
                     sim5.1$"4",sim5.1$"5",sim5.1$"6"))
```

Desorden de los datos

```
s<-sample(size)

data.1.n<-data.1[,s]
```

CPCA

```
CPCA_order<-function(E1,E2){
  e1<-E1/sqrt(E1^2+E2^2)
  e2<-E2/sqrt(E1^2+E2^2)
  return(atan2(e2,e1))
}
```

```
p1<-PCA(t(data.1.n[]),scale.unit = FALSE,graph=FALSE)
summary(p1)
pred<-predict(p1,newdata=t(data.1.n[]))$coord[,1:2]
E1<-pred[,1]
E2<-pred[,2]
order1.1<-CPCA_order(E1,E2)+pi

#sincronización
tu3<-(pi/2)+2*atan(tan(-3*pi/4)/0.9)
peek_3<-which.max(fitFMM(data.1.n[3,order(order1.1)],
                         timePoints = sort(order1.1))@fittedValues)
order1.1s<-order1.1+(-sort(order1.1)[peek_3]+tu3)

order1.1<-sapply(order1.1s,normalize_angle)
#order1.1<-2*pi-order1.1

plot(sort(order1.1),data.1.n[1,order(order1.1)])
plot(sort(order1.1),data.1.n[2,order(order1.1)])
plot(sort(order1.1),data.1.n[3,order(order1.1)])
plot(sort(order1.1),data.1.n[4,order(order1.1)])
plot(sort(order1.1),data.1.n[5,order(order1.1)])
plot(cos(order1.1),sin(order1.1))
```

Red neuronal

```
write.csv(data.1.n,file="eigengenes.csv",row.names = FALSE)#500
```

Se ejecuta en matlab el código de la red neuronal y se vuelve a leer en R

```
order1.2<-as.numeric(read.csv("CPCA.csv",header=FALSE))+pi
#sincronización
peek_3<-which.max(fitFMM(data.1.n[3,order(order1.2)],
                         timePoints = sort(order1.2))@fittedValues)
order1.2s<-order1.2+(-sort(order1.2)[peek_3]+tu3)

order1.2<-sapply(order1.2s,normalize_angle)
#order1.2<-2*pi-order1.2

plot(sort(order1.2),data.1.n[1,order(order1.2)])
plot(sort(order1.2),data.1.n[2,order(order1.2)])
plot(sort(order1.2),data.1.n[3,order(order1.2)])
plot(sort(order1.2),data.1.n[4,order(order1.2)])
plot(sort(order1.2),data.1.n[5,order(order1.2)])
plot(cos(order1.2),sin(order1.2))
```

TSP

```
source("D:/Universidad/TFG-Estadistica/Codigo y articulos/funcionTSP.R")
```

```
ordenTsp <-
  TSP_Euc_v6(
    datos = data.1.n,
    dist_type = "Euclidea",
    datos2p = mPrueba,
    pesosE = FALSE,
    pesosMSE = TRUE,
    pesosAdd = FALSE,
    unPeriodo = TRUE,
    pesos = rep(1, ncol(datos)),
    centrar = FALSE,
    intentos = 3,
    onlyHeuristica2 = FALSE
  )
```

```
order1.3<-ordenTsp[[1]]
#sincronización
peek_3<-which.max(fitFMM(data.1.n[3,order1.3])@fittedValues)
tu3.3<-round(tu1/(2*pi)*500)
order1.3<-order1.3[c(393:500,1:392)]
order1.3<-rev(order1.3)
plot(data.1.n[1,order1.3])
plot(data.1.n[2,order1.3])
plot(data.1.n[3,order1.3])
plot(data.1.n[4,order1.3])
plot(data.1.n[5,order1.3])
```

Medida de comparación de los modelos

```
fit1<-apply(data.1.n[1:4,order(order1.1)],1,get_fit_sim)
fit2<-apply(data.1.n[1:4,order(order1.2)],1,get_fit_sim)

resid1<-mean(sapply(c(1:4),function(i){abs(fit1[[i]]$y-data.1[i,])}))
resid2<-mean(sapply(c(1:4),function(i){abs(fit2[[i]]$y-data.1[i,])}))
resid1;resid2;
```

Carga de datos

Samuel Bassols

2023-06-02

Lectura de datos

```
load(paste("D:/Universidad/TFG-Estadistica/OneDrive_1_9-15-2022/datos/",
           "covDB.RData",sep=""))
load(paste("D:/Universidad/TFG-Estadistica/OneDrive_1_9-15-2022/datos/",
           "db.RData",sep=""))
dim(db)
load(paste("D:/Universidad/TFG-Estadistica/OneDrive_1_9-15-2022/datos/",
           "fullGenes.RData",sep=""))
length(fullGenes)
load(paste(file = "D:/Universidad/TFG-Estadistica/OneDrive_1_9-15-2022/datos/",
           "dbCoreG.RData",sep=""))
#54 genes core
core <-
  read.csv(file = "D:/Universidad/TFG-Estadistica/core2.csv",
           header = FALSE)[, 1]
```

```
tTissue<-table(covDB[,"SMTSD"])

tissue40<-sort(tTissue[tTissue>40],decreasing = TRUE)
```

Tejidos con mas de 600 mustras

```
d<-as.data.frame(tissue40)

d[d$Freq>600,]
```

Datos para músculo- esqueletico

```
covDBMS<-covDB[covDB$SMTSD=="Muscle - Skeletal",]
saveRDS(db[,covDBMS$SAMPID],
        file= "D:/Universidad/TFG-Estadistica/datos/genes_unordered1.rds")
saveRDS(dbCoreG[,covDBMS$SAMPID],
        file= "D:/Universidad/TFG-Estadistica/datos/genes_unordered_core1.rds")
saveRDS(covDBMS[,c("AGE","SEX")],
        file= "D:/Universidad/TFG-Estadistica/datos/covAS.rds")
```

Preprocesado

Samuel Bassols

2023-06-02

```
source("D:/Universidad/TFG-Estadistica/FuncionesTFGEST.R")
```

Lectura de datos

```
genes_unordered<-readRDS(
  "D:/Universidad/TFG-Estadistica/datos/genes_unordered1.rds")
genes_unordered_core<-readRDS(
  "D:/Universidad/TFG-Estadistica/datos/genes_unordered_core1.rds")
core<-read.csv(file="core2.csv",header = FALSE)[,1]
```

Eliminar genes con valores 0 y transformar los datos mediante log

```
filter_0<-remove_zeroes(genes_unordered,threshold = 0.3)

genes_unordered<-genes_unordered[filter_0,]

genes_unordered<-log2(genes_unordered+1)
genes_unordered_core<-log2(genes_unordered_core+1)
genes_trimmed_mean<-apply(genes_unordered,1,upper_trim_mean,trim=0.1)
plot(density(genes_trimmed_mean))
```

Graficos para el threshold en la media recortada

```
seque<-seq(0,5,by=0.1)
s1<-sapply(seque,
           test_threshold,d=genes_trimmed_mean,
           gene_data=genes_unordered,core=core)
plot(seque,s1[1,]) #proporcion que tiene la media por debajo
plot(seque,s1[2,]) #proporcion de cores [54] seleccionado
```

Eliminar genes mediante la media recortada

```
selection_trimmed<-genes_trimmed_mean>0.5
table(selection_trimmed)
genes_unordered<-genes_unordered[selection_trimmed,]
```

Normalizar y guardar los datos

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

saveRDS(genes_unordered,
        file= "D:/Universidad/TFG-Estadistica/datos/genes_unordered2.rds")
saveRDS(genes_unordered_core,
        file= "D:/Universidad/TFG-Estadistica/datos/genes_unordered_core2.rds")
```

Ordenación de las muestras

Samuel Bassols

2023-06-02

```
library(FactoMineR)
library(FMM)
source("D:/Universidad/TFG-Estadistica/FuncionesTFGEST.R")
```

Lectura de datos

```
coreG<-c("PER1" , "PER2" , "PER3" , "CRY1" ,
         "CRY2" , "ARNTL" ,"CLOCK" ,"NR1D1", "RORA" ,
         "DBP" ,  "TEF"   , "STAT3")
core<-read.csv(file="core2.csv",header = FALSE)[,1]
genes_unordered<-readRDS(
  "D:/Universidad/TFG-Estadistica/datos/genes_unordered2.rds")
genes_unordered_core<-readRDS(
  "D:/Universidad/TFG-Estadistica/datos/genes_unordered_core2.rds")
covAS<-readRDS("D:/Universidad/TFG-Estadistica/datos/covAS.rds")
```

Ordenación por componentes principales

```
p1<-PCA(t(genes_unordered_core),scale.unit = FALSE,graph=FALSE)
if(p1$eig[2,2]<25) warning(
  "Poca variabilidad explicada por los dos primeros eigengenes")
pred<-predict(p1,newdata=t(genes_unordered_core))$coord[,1:3]
E1<-pred[,1]
E2<-pred[,2]
E3<-pred[,3]
plot(E1,E2)
order1<-CPCA_order(E1,E2)
order1<-order1+pi #[0,2*pi]
```

Eigengenes ordenados

```
E1.order<-E1[order(order1)]
E2.order<-E2[order(order1)]
plot(sort(order1),E1.order)
plot(sort(order1),E2.order)
plot(cos(order1),sin(order1))
```

Datos ordenados

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

```
for( i in coreG){
  plot(angle_order1*24/(2*pi),genes_ordered_core1[i,],
       xlab = "h",ylab = "expression",xaxt='n',yaxt='n',xlim=c(0,24))
  title(main=i)
  axis(1,at = c(0, 12, 24))
}
```

Eliminación de outliers y ordenación

```
no_outliers<-apply(apply(genes_ordered_core1,1,FMMOutliers),1,and_logical)
no_outliers<-no_outliers[order(order(order1))]
genes_unordered<-genes_unordered[,no_outliers]
genes_unordered_core<-genes_unordered_core[,no_outliers]
covAS<-covAS[order(order1),][no_outliers,]
table(no_outliers)

#volvemos a normalizar
genes_unordered<-t(apply(genes_unordered, 1, normalize))
genes_unordered_core<-t(apply(genes_unordered_core, 1, normalize))
#volvemos a ordenar
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)
order2<-order2+pi
genes_ordered2<-genes_unordered[,order(order2)]
genes_ordered_core2<-genes_unordered_core[,order(order2)]
angle_order2<-sort(order2)
```

```
for( i in coreG){
  plot(angle_order2*24/(2*pi),genes_ordered_core2[i,],
       xlab = "h",ylab = "expression",xaxt='n',yaxt='n',xlim=c(0,24))
  title(main=i)
  axis(1,at = c(0, 12, 24))
  print(fitFMM(genes_ordered_core2[i,],timePoints = angle_order2)@R2)
}
```

Sincronización de los genes

```
peek_arntl<-which.max(
  fitFMM(genes_ordered_core2["ARNTL",],timePoints = angle_order2)@fittedValues)
```

```
order2s<-order2+(-angle_order2[peek_arntl]+pi)
order2s<-sapply(order2s,normalize_angle)
angle_order2s<-sort(order2s)
plot(angle_order2,genes_unordered_core["ARNTL",order(order2)])
plot(angle_order2,genes_unordered_core["DBP",order(order2)])
genes_ordered_core2<-genes_unordered_core[,order(order2s)]

peek_per3<-which.max(
  fitFMM(genes_ordered_core2["PER3",],timePoints = angle_order2s)@fittedValues)

if(angle_order2s[peek_per3]>pi){
  order2s<-2*pi-order2s
}



plot(angle_order2s,genes_unordered_core["ARNTL",order(order2s)])
plot(angle_order2s,genes_unordered_core["DBP",order(order2s)])


order2<-order2s
genes_ordered2<-genes_unordered[,order(order2)]
genes_ordered_core2<-genes_unordered_core[,order(order2)]
angle_order2<-sort(order2)


peek_per1<-which.max(
  fitFMM(genes_ordered_core2["PER1",],timePoints = angle_order2)@fittedValues)
peek_per2<-which.max(
  fitFMM(genes_ordered_core2["PER2",],timePoints = angle_order2)@fittedValues)

if(angle_order2[peek_per1]>pi||angle_order2[peek_per2]>pi){
  warning(paste("Es posible que el orden sea incorrecto,",
                " los genes PER deberían de estar sincronizados",sep=""))
}
```

Guardar los datos

```
saveRDS(order2,"D:/Universidad/TFG-Estadistica/datos/order.rds")
saveRDS(genes_unordered,
        file= "D:/Universidad/TFG-Estadistica/datos/genes_unordered3.rds")
saveRDS(genes_unordered_core,
        file= "D:/Universidad/TFG-Estadistica/datos/genes_unordered_core3.rds")
saveRDS(covAS,file= "D:/Universidad/TFG-Estadistica/datos/covAS2.rds")
```

Ajuste del modelo FMM para cada gen

Samuel Bassols

2023-06-02

```
library(FMM)
library(parallel)
```

Lectura de datos

```
order_obs<-readRDS("D:/Universidad/TFG-Estadistica/datos/order.rds")
genes_unordered<-readRDS("D:/Universidad/TFG-Estadistica/datos/genes_unordered3.rds")
genes_unordered_core<-readRDS("D:/Universidad/TFG-Estadistica/datos/genes_unordered_core3.rds")
genes_ordered<-genes_unordered[,order(order_obs)]
genes_ordered_core<-genes_unordered_core[,order(order_obs)]
angle_order<-sort(order_obs)
```

Ajuste FMM

```
n.cores <- detectCores()
clust <- makeCluster(n.cores)

clusterExport(clust, "fitFMM")

gene_fitted<-parApply(clust, genes_ordered, 1,fitFMM,timePoints=angle_order)
```

Guardado de datos

```
saveRDS(gene_fitted,"D:/Universidad/TFG-Estadistica/datos/FMM_genes.rds")
```

Clasificación y detección de genes rítmicos

Samuel Bassols

2023-06-02

```
library(FMM)
library(fdapace)
library(circular)
library(modi)
library(fpc)
source("D:/Universidad/TFG-Estadistica/FuncionesTFGEST.R")
```

Lectura de datos

```
order_obs<-readRDS("D:/Universidad/TFG-Estadistica/datos/order.rds")
genes_unordered<-readRDS(
  "D:/Universidad/TFG-Estadistica/datos/genes_unordered3.rds")
genes_unordered_core<-readRDS(
  "D:/Universidad/TFG-Estadistica/datos/genes_unordered_core3.rds")
genes_ordered<-genes_unordered[,order(order_obs)]
genes_ordered_core<-genes_unordered_core[,order(order_obs)]
angle_order<-sort(order_obs)
FMM_genes<-readRDS("D:/Universidad/TFG-Estadistica/datos/FMM_genes.rds")
covAS<-readRDS("D:/Universidad/TFG-Estadistica/datos/covAS2.rds")
```

Centrado de los datos

```
filter1<-sapply(FMM_genes,function(fit){fit@R2>0.4})
table(filter1)
FMM_genes<-FMM_genes[filter1]
genes_ordered<-genes_ordered[filter1,]


filter2<-sapply(FMM_genes,function(fit){fit@omega>0.1})

FMM_genes<-FMM_genes[filter2]
genes_ordered<-genes_ordered[filter2,]
```

```
genes_predict<-t(sapply(FMM_genes,function(fit) fit@fittedValues))
genes_predict<-t(apply(genes_predict,1,centre_mean))
```

FPCA de los genes

```
set.seed("2023")
K<-10
genes_functional <- MakeFPCAInputs(tVec=angle_order, yVec=genes_predict)

fpca_1 <- FPCA(genes_functional$Ly, genes_functional$Lt)

c_1<-fpca_1$xiEst[,1:2]
```

Clustering k-meadoids

```
cluster_1<-pamk(c_1, krange=5:7,usepam=FALSE)
clusters<-cluster_1$pamobject$clustering
K<-cluster_1$nc
```

gen medio de cada cluster escalado por el R2

```
FMM_values<-data.frame(matrix(ncol = 7, nrow = 0))
for(i in 1:length(clusters)){
  FMM_values<-rbind(
    FMM_values,c(unlist(coef(FMM_genes[[i]])),clusters[i],FMM_genes[[i]]@R2))
}
colnames(FMM_values)<-c("M","A","alpha","beta","omega","cluster","R2")
FMM_values$M<-rep(0,length(FMM_genes))
```

```
mean_valuesR2<-data.frame(matrix(ncol = 5, nrow = 0))
sd_valuesR2<-data.frame(matrix(ncol = 5, nrow = 0))
for(i in 1:K){
  clust<-FMM_values$cluster==i
  weightsR2<-FMM_values[clust,]$R2
  M<-weighted.mean(FMM_values[clust,]$M,weightsR2)
  Md<-sqrt(weighted.var(FMM_values[clust,]$M,weightsR2))

  A<-weighted.mean(FMM_values[clust,]$A,weightsR2)
  Ad<-sqrt(weighted.var(FMM_values[clust,]$A,weightsR2))

  alphad<-weighted_sd_circ(FMM_values[clust,]$alpha,w=weightsR2)
  alpha<-circular(FMM_values[clust,]$alpha,modulo="2pi")
  alpha<-weighted.mean.circular(alpha%% (2*pi),weightsR2)

  beta<-circular(FMM_values[clust,]$beta,modulo = "2pi")
  betad<-weighted_sd_circ(FMM_values[clust,]$beta,w=weightsR2)
  beta<-weighted.mean.circular(beta%% (2*pi),weightsR2)

  omega<-weighted.mean(FMM_values[clust,]$omega,weightsR2)
  omegad<-sqrt(weighted.var(FMM_values[clust,]$omega,weightsR2))
  mean_valuesR2<-rbind(mean_valuesR2,c(M,A,alpha,beta,omega))
```

```
  sd_valuesR2<-rbind(sd_valuesR2,c(Md,Ad,alphad,betad,omegad))
  gFMM<-generateFMM(M,A,alpha=alpha,beta=beta,omega,
                    plot = FALSE,length.out = 500,outvalues = TRUE)
  plot(gFMM$t,gFMM$y,ylim = c(- 1.2, 1.2))
  title(table(clusters)[i])
}
colnames(sd_valuesR2)<-c("M","A","alpha","beta","omega")
mean_valuesR2
colnames(sd_valuesR2)<-c("M","A","alpha","beta","omega")
sd_valuesR2
```

Guardado de datos de un cluster

```
cluster<-1

saveRDS(FMM_genes[clusters==cluster],"D:/Universidad/TFG-Estadistica/datos/cluster1.rds")
```

Enriquecimiento para un cluster

2023-06-15

Lectura de datos

```
genes<-readRDS("D:/Universidad/TFG-Estadistica/datos/cluster1.rds")
```

Matriz con los datos parámetros/gen

```
pCluster1<-t(sapply(1:length(genes),function(i)
  c(genes[[i]]@M,genes[[i]]@A,genes[[i]]@alpha,genes[[i]]@beta,genes[[i]]@omega,
    genes[[i]]@R2,sigma=sd(genes[[i]]@data-genes[[i]]@fittedValues,na.rm=TRUE))))
colnames(pCluster1)<-c("M","A","alpha","beta","omega","R2","sigma")
rownames(pCluster1)<-names(genes)
pCluster1<-as.data.frame(pCluster1)
```

Anotación

```
codEnsemble<-rownames(pCluster1)
codEnsemble<-substr(codEnsemble,start=1,stop=15)


if(!("clusterProfiler" %in% installed.packages()))
  BiocManager::install("clusterProfiler")
library("org.Hs.eg.db")
library("clusterProfiler")

# R2 usado para ordenar los datos
gene_list <- pCluster1$R2

# nombres de los elementos
names(gene_list) <- codEnsemble
# eliminamos de la lista los NA
gene_list<-na.omit(gene_list)
# ordenamos de mayor a menor
gene_list <- sort(gene_list, decreasing = TRUE)
head(gene_list,n=5)
```

Enriquecimietno GSEA

```
gse <- gseGO(geneList=gene_list,
             ont ="ALL",
             keyType = "ENSEMBL",
             nPerm = 10000,
             minGSSize = 3,
             maxGSSize = 800,
             pvalueCutoff = 0.05,
             OrgDb = org.Hs.eg.db,
             pAdjustMethod = "none")
```

Resultados para la categoría procesos biologógicos BP

```
gse.BP<-gse


gse.BP@result<-gse@result[gse@result[,1]=="BP",]
```

```
library(enrichplot)
library(ggupset)
enrichplot::treeplot(enrichplot::pairwise_termsim(gse.BP))
```

CPCA.m

```
rng(2023)
M = readmatrix( "../eigengenes.csv" );
%%
%%
[c,net,network]=nlpca(M,1,'type', 'inverse' ,...
                          'circular', 'yes'    ,...
                          'max_iteration', 500 );
%%
writematrix(c,"../CPCA.csv")
```