FMM: An R Package for Modeling Rhythmic Patterns in Oscillatory Systems

by Itziar Fernández, Alejandro Rodríguez-Collado, Yolanda Larriba, Adrián Lamela, Christian Canedo and Cristina Rueda

Abstract This paper is dedicated to the R package **FMM** which implements a novel approach to describe rhythmic patterns in oscillatory signals. The frequency modulated Möbius (FMM) model is defined as a parametric signal plus a Gaussian noise, where the signal can be described as a single or a sum of waves. The FMM approach is flexible enough to describe a great variety of rhythmic patterns. The **FMM** package includes all required functions to fit and explore single and multi-wave FMM models, as well as a restricted version that allows equality constraints between parameters representing a priori knowledge about the shape to be included. Moreover, the **FMM** package can generate synthetic data and visualize the results of the fitting process. The potential of this methodology is illustrated with examples of such biological oscillations as the circadian rhythm in gene expression, the electrical activity of the heartbeat and the neuronal activity.

Introduction

Oscillations naturally occur in a multitude of physical, chemical, biological, and even economic and social processes. Periodic signals appear, for example, during the cell-cycle, in biological time-keeping processes, in human heartbeats, in neuronal signals, in light emissions from certain types of stars, or in business cycles in economics, among many others. Three features typically describe the periodic nature of the oscillatory motion: period, amplitude and phase. The period is the time required for one complete oscillation. Within a period, a sum of monocomponent models, characterized by the phase and amplitude parameters, can be used to describe the rhythmic pattern of a signal (Boashash, 2016). By varying the number of monocomponents and considering phase and amplitude parameters as fixed or variable, a large number of rhythmic signal representations can be found.

One of the most popular representations of oscillating signals is the Fourier decomposition (FD): a multicomponent representation with a fixed amplitude parameter. Its monocomponent version, the cosinor model (COS) (Cornelissen, 2014), is widely used, in particular in chronobiology, with acceptable results when a sinusoidal shape response within a period is expected. Due to its widespread use, many software utilities are available. Particularly in R, the estimation of a COS model can be performed using **cosinor** (Sachs, 2014) and **cosinor2** packages (Mutak, 2018). In addition, other packages from widely differing areas of knowledge have specific functions for fitting COS models. Such is the case of, for example, the function `CATCosinor` in the **CATkit** package (Gierke et al., 2018), which implements tools for periodicity analysis; the function `cosinor` in the **psych** package (Revelle, 2021), dedicated to personality and psychological research; or the function `cosinor` contained in a recent package, **card** (Shah, 2020), which is dedicated to the assessment of the regulation of cardiovascular physiology. Recently, it has also been implemented in other languages such as CosinorPy, a cosinor python package (Moskon, 2020). The COS model is easy to use and interpret with symmetrical patterns. However, asymmetric shapes are not captured properly by COS. When the waveform is nonsinusoidal, the use of multiple components analysis to fit a model consisting of a sum of several periodical functions is recommended. However, the multicomponent FD models, developed to provide flexibility from COS, often require the use of a large number of components resulting in serious overfitting issues.

In recent years, alternative methods, mostly nonparametric statistical methods, have been developed and used for analyzing rhythmicity, especially in biological data sets. Some very popular ones, such as the JTK_CYCLE (Hughes et al., 2010), wrongly assume that any underlying rhythms have symmetric waveforms. Others, such as RAIN (Thaben and Westermark, 2014), designed to detect more diverse wave shapes including asymmetric patterns, are not focused on modeling but on detecting rhythmic behavior in sets of data. Thus, they are not useful to describe the underlying oscillatory phenomena. The proliferation of methodology in this field has been accompanied by software developments. This is the case, for example, of the **DiscoRhythm** R package (Carlucci et al., 2020), very recently available on Bioconductor with a web interface based on the R **Shiny** platform (Chang et al., 2021). This tool allows four popular approaches to be used, including the COS model and JTK_CYCLE, to discover biological rhythmicity. Another recent example is the **circacompare** (Parsons et al., 2020), an R package implemented for modeling cosinusoidal curves by nonlinear regression. Hosted on GitHub, we can also find the **LimoRhyde** R package (`https://github.com/hugheylab/limorhyde`) for the differential analysis of rhythmic transcriptome data, based on fitting linear models (Singer and Hughey, 2019).

Motivated by the need for a flexible, interpretable and parametric methodology to fit rhythmic patterns, our research group recently proposed the frequency modulated Möbius (FMM) model (Rueda et al., 2019). The FMM is an additive nonlinear parametric regression model capable of adapting to nonsinusoidal shapes and whose parameters are easily interpretable. The single component model has been shown to successfully fit data as diverse as circadian clock signals, hormonal levels data or light data from distant stars. In addition, for more complex oscillatory signals, a multicomponent model of order m, denoted as FMM$_m$, which includes m single FMM components, can be used. This is, for example, the case for describing electrocardiography (ECG) signals. The FMM$_{ecg}$ signal, presented in Rueda et al. (2021b), is defined as the combination of five single FMM components. Another interesting area where the FMM approach has already shown its usefulness is in electrophysiological neuroscience. Specifically, we have proposed FMM methodology for modeling neuronal action potential (AP) curves, oscillating signals that measure the difference between the electrical potential inside and outside the cell (see Rueda et al., 2021c; Rodríguez-Collado and Rueda, 2021a). An FMM$_2$ model provides an accurate fitting for a single AP curve; whereas series of AP curves with similar repetitive spikes can be efficiently fitted by the FMM$_{ST}$ model, a restricted version of the multicomponent FMM model.

In this work we introduce the **FMM** package (Fernández et al., 2021), programmed in R and available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=FMM`. The package implements all required functions to fit and explore single and multicomponent FMM models, as well as a restricted multicomponent version. In addition, the **FMM** package provides functions to generate synthetic data and visualize the results of the fitted model. Furthermore, its use is illustrated in the aforementioned applications. The remainder of this paper is organized as follows: the next section provides a brief overview of both mono and multicomponent FMM models, as well as the FMM$_m$ model with equality constraints. The section follows is dedicated to the implementation details of the **FMM** package. After that, through a simulated example, the basic usage of the package is introduced, including the data generation and the fitting, as well as the visualization of the results. Then, the **FMM** package performance is shown through three application areas governed by oscillatory systems: chronobiology, ECG and neuroscience. Finally, a summary is provided.

Frequency modulated Möbius (FMM) model

FMM is a new approach to describe a great variety of rhythmic patterns in oscillatory signals as the composition of several additive components. In this section an overview of the FMM approach is provided. All the methodological details that justify the mathematical formulation of the FMM models are given in Rueda et al. (2019).

At the time point t, a single FMM wave is defined as $W(t; \upsilon) = A\cos(\phi(t; \alpha, \beta, \omega))$ where $\upsilon = (A, \alpha, \beta, \omega)'$, $A \in \Re^+$ represents the wave amplitude and,

$$\phi(t; \alpha, \beta, \omega) = \beta + 2\arctan\left(\omega \tan\left(\frac{t-\alpha}{2}\right)\right) \tag{1}$$

the wave phase. The phase angle ϕ of an FMM wave is defined using the Möbius link (see Downs and Mardia, 2002; Kato et al., 2008) rather than the linear link function as in the COS model. The Möbius link provides much more flexibility to describe nonsinusoidal patterns. Without loss of generality, we assume that the time point $t \in [0, 2\pi]$. Otherwise it can be transformed into $t' \in [t_0, T + t_0]$ by $t = \frac{(t'-t_0)2\pi}{T}$.

Each of the four parameters of an FMM wave characterizes some aspect of a rhythmic pattern. A describes the amplitude of the signal, while α, β and ω describe the wave phase. $\alpha \in [0, 2\pi]$ is a translation parameter and a wave location parameter in the real space, whereas $\beta \in [0, 2\pi]$ and $\omega \in [0, 1]$ describe the wave shape. To be precise, assuming $\alpha = 0$, the unimodal symmetric waves are characterized by values of β close to 0, π or 2π. When $\beta = \frac{\pi}{2}$ or $\beta = \frac{3\pi}{2}$, extreme asymmetric patterns are described. Moreover, a value of ω close to zero describes an extreme spiked wave and, as ω value increases, the pattern is increasingly smoother. When $\omega = 1$, a sinusoidal wave is described and the FMM model matches the COS model where $\varphi = \beta - \alpha$ is the acrophase parameter.

Two important features of a wave are the peak and trough, defined as the highest and lowest points above and below the rest position, respectively. In many applications, the peak and trough times could be very useful tools to extract practical information of a wave, since they capture important aspects of the dynamics. These two interesting parameters can be directly derived from the main parameters of an FMM wave as,

$$t^U = \alpha + 2\arctan\left(\frac{1}{\omega}\tan\left(-\frac{\beta}{2}\right)\right) \tag{2}$$
$$t^L = \alpha + 2\arctan\left(\frac{1}{\omega}\tan\left(\frac{\pi-\beta}{2}\right)\right)$$

where t^U and t^L denote the peak and trough times, respectively.

Monocomponent FMM model

Let $X(t_i)$, $t_1 < t_2 < \cdots < t_n$ be the vector of observations. The monocomponent FMM model is defined as follows:

$$X(t_i) = M + W(t_i; v) + e(t_i), \quad i = 1, \ldots, n \tag{3}$$

where $M \in \Re$ is an intercept parameter describing the baseline level of the signal, $W(t_i; v)$ is an FMM wave, and it is assumed that the errors $e(t_i)$ are independent and normally distributed with zero mean and a common variance σ^2.

Estimation algorithm

A two-step algorithm to estimate monocomponent FMM model parameters is proposed. We now describe the substantial details of each stage of the algorithm.

Step 1: Initial parameter estimation. A two-way grid search over the choice of (α, ω) parameters is performed. For each pair of (α, ω) fixed values, the estimates for M, A and β are obtained by solving a least square problem as detailed below.

The model for a single FMM component can be written as:

$$X(t_i) = M + A\cos(t_i^* + \varphi) + e(t_i) \tag{4}$$

where $t_i^* = \alpha + 2\arctan\left(\omega\tan\left(\frac{t_i-\alpha}{2}\right)\right)$, $\varphi = \beta - \alpha$, and $e(t_i) \sim N(0, \sigma^2)$ for $i = 1, \ldots, n$.

Using trigonometric angle sum identity, the model can be rewritten as:

$$X(t_i) = M + \delta z_i + \gamma w_i + e(t_i) \tag{5}$$

where $\delta = A\cos(\varphi)$, $\gamma = -A\sin(\varphi)$, $z_i = \cos(t_i^*)$ and $w_i = \sin(t_i^*)$.

Since α and ω are fixed, the estimates for M, δ and γ are obtained by minimizing the residual sum of the squares (RSS),

$$RSS = \sum_{i=1}^{n}\left(X(t_i) - (\hat{M} + \hat{\delta} z_i + \hat{\gamma} w_i)\right)^2 \tag{6}$$

And the estimates for M, A and β are straightforward to derive as follows,

$$\hat{M} = \bar{X} - \hat{\delta}\sum_{i=1}^{n} z_i - \hat{\gamma}\sum_{i=1}^{n} w_i \tag{7}$$
$$\hat{A} = \sqrt{\hat{\delta}^2 + \hat{\gamma}^2} \tag{8}$$
$$\hat{\beta} = \alpha + \varphi \tag{9}$$

The best combination of (α, ω) values, with the lowest RSS, is retained and the corresponding estimates are the initial parameter estimation values.

Step 2: Optimization. In the second step, the Nelder-Mead optimization method (Nelder and Mead, 1965) is used to obtain the final FMM parameter estimates that minimize the RSS.

Multicomponent FMM model

A multicomponent FMM model of order m, denoted by FMM$_m$, is defined as

$$X(t_i) = M + \sum_{J=1}^{m} W(t_i; v_J) + e(t_i) \tag{10}$$
$$t_1 < t_2 < \cdots < t_n; i = 1, \ldots, n$$

where $W(t_i; v_J)$, hereinafter denoted by $W_J(t_i)$, is the Jth FMM wave and,

- $M \in \Re$
- $v_J = (A_J, \alpha_J, \beta_J, \omega_J)' \in \Re^+ \times [0, 2\pi] \times [0, 2\pi] \times [0, 1]; J = 1, \ldots, m$
- $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_m \leq \alpha_1$
- $(e(t_1), \ldots, e(t_n))' \sim N_n(0, \sigma^2 I_n)$

Model adequacy

The goodness of fit of an FMM model is measured with the R^2 statistic that represents the proportion of the variance explained by the model out of the total variance, that is:

$$R^2 = 1 - \frac{\sum_{i=1}^n \left(X(t_i) - \hat{X}(t_i)\right)^2}{\sum_{i=1}^n \left(X(t_i) - \bar{X}\right)^2} \tag{11}$$

where $\hat{X}(t_i)$ represents the fitted value at $t_i, i = 1, \ldots, n$.

Estimation algorithm

An iterative backfitting algorithm is proposed to derive estimates for the FMM parameters. Let $\hat{W}_J^{(k)}(t_i)$ denote the fitted values from the Jth FMM wave at $t_i, i = 1, ..., n$ in the kth iteration. The algorithm is structured as follows:

1. Initialize. Set $\hat{W}_1^{(0)}(t_i) = \cdots = \hat{W}_m^{(0)}(t_i) = 0$.
2. Backfitting step. For $J = 1, \ldots, m$, calculate

$$r_J^{(k)}(t_i) = X(t_i) - \sum_{I<J} \hat{W}_I^{(k)}(t_i) - \sum_{I>J} \hat{W}_I^{(k-1)}(t_i); I = 1, \ldots, m \tag{12}$$

   and fit a monocomponent FMM model to $r_J^{(k)}(t_i)$ obtaining $\hat{\alpha}_J^{(k)}$, $\hat{\beta}_J^{(k)}$, $\hat{\omega}_J^{(k)}$ and $\hat{W}_J^{(k)}(t_i)$.
3. Repeat the backfitting step until the stopping criterion is reached. The stopping criterion is defined as the difference between the explained variability in two consecutive iterations: $R_k^2 - R_{k-1}^2 \leq C$, where R_k^2 (defined in Equation 11) is the proportion of variance explained by the model in the kth iteration and C a constant.
4. $\hat{M}$ and $\hat{A}_J$ are derived by solving

$$\min_{M \in \Re; A_J \in \Re^+} \sum_{i=1}^n \left(X(t_i) - M - \sum_{J=1}^m A_J \cos\left(\hat{\phi}_J(t_i)\right) \right)^2 \tag{13}$$

   where $\hat{\phi}_J(t_i) = \phi(t_i; \hat{\alpha}_J, \hat{\beta}_J, \hat{\omega}_J)$ defined in Equation 1.

Restricted multicomponent FMM model

Modeling signals with repetitive shape-similar waves can be very useful in some applications (see Rodríguez-Collado and Rueda, 2021a). In order to obtain more efficient estimators, equality constraints are imposed on the β and ω parameters of an FMM$_m$ model. In particular, we add d blocks of restrictions:

$$\begin{array}{rcl} \beta_1 = \cdots = \beta_{m_1} & & \omega_1 = \cdots = \omega_{m_1} \\ \beta_{m_1+1} = \cdots = \beta_{m_2} & & \omega_{m_1+1} = \cdots = \omega_{m_2} \\ \cdots & & \\ \beta_{m_{d-1}+1} = \cdots = \beta_{m_d} & & \omega_{m_{d-1}+1} = \cdots = \omega_{m_d} \end{array} \tag{14}$$

The parameter estimation problem is solved by an adaptation of the standard procedure.

FMM$_m$ estimation algorithm with restrictions on the β parameters

Given the unrestricted estimates obtained in step 3, the estimates for $\beta_1, \beta_{m_1+1}, \ldots, \beta_{m_{d-1}+1}$ under equality restrictions (Equation 14) are computed as follows:

$$
\begin{aligned}
\hat{\beta}_J^* &= \text{angularMean}\left(\hat{\beta}_1, \ldots, \hat{\beta}_{m_1}\right) && J = 1, \ldots, m_1 \\
\hat{\beta}_J^* &= \text{angularMean}\left(\hat{\beta}_{m_1+1}, \ldots, \hat{\beta}_{m_2}\right) && J = m_1 + 1, \ldots, m_2 \\
&\ldots \\
\hat{\beta}_J^* &= \text{angularMean}\left(\hat{\beta}_{m_{d-1}+1}, \ldots, \hat{\beta}_{m_d}\right) && J = m_{d-1} + 1, \ldots, m_d
\end{aligned}
$$

Then, the algorithm continues to the next step.

FMM$_m$ estimation algorithm with restrictions on the ω parameters

When constraints for the ω parameters are incorporated, the grid search for the different ω values is outside the backfitting loops. When the number of blocks is large, the estimation procedure can be computationally unaffordable. In order to reduce the execution time, a two-nested backfitting algorithm is proposed. In the outer backfitting loop, a block is fitted. In the inner loop, the FMM waves belonging to the same block are estimated. This procedure generates a close to optimal solution and is a less computationally expensive alternative.

FMM package: Implementation details

The **FMM** code makes use of the **doParallel** package (Corporation and Weston, 2020) to embed parallelization for the fitting process. Several utilities from the **ggplot2** (Wickham, 2016) and **RColorBrewer** (Neuwirth, 2014) packages are occasionally necessary for the visualization of the fitted models.

The implementation of **FMM** is divided into four main functionalities described in the next four sections: the fitting of the FMM models, the new `S4` object of class `"FMM"`, the graphical visualization of the fittings and the simulation of synthetic data.

Some general details about the functions contained in the **FMM** package are shown in Table 1.

Function	Description
Fitting function	
`fitFMM(vData,timePoints,nback,...)`	Estimates an FMM$_{nback}$ model to `vData` observed at `timePoints`.
Utility functions	
`plotFMM(objFMM,...)`	Graphically displays an object of class `"FMM"`.
`generateFMM(M,A,alpha,beta,omega,...)`	Simulates values from an FMM model with parameters ($M =$ `M`, $A =$ `A`, $\alpha =$ `alpha`, $\beta =$ `beta`, $\omega =$ `omega`).
`getFMMPeaks(objFMM,...)`	Estimates peak and trough times, together with signal values at those times, for each FMM wave.
`extractWaves(objFMM)`	Extracts individual contribution to the fitted values of each FMM wave.
Standard methods for objects of class `"FMM"`	
`summary()`, `show()`, `coef()`, `fitted()`	

Table 1: Summary of the fitting, utility functions and standard methods implemented in **FMM** package.

Fitting an FMM model

An FMM model can be fitted using the main function `fitFMM()`. The description and default values of its inputs arguments are shown in Table 2.

The fitting function `fitFMM()` requires the `vData` input argument, which contains the data to be fitted. Two other arguments can be used to control a basic fitting: `timePoints`, which contains the specific time points of the single period; and `nback`, with the number of FMM components to be fitted. For some applications, such as the study of circadian rhythms, data are collected over multiple periods. This information is received by the `fitFMM()` function through the input argument `nPeriods`. When `nPeriods>1`, the FMM fitting is carried out by averaging the data collected at each time point across all considered periods.

Argument	Default value	Description
`vData`	no default value	A `"numeric"` vector containing the data to be fitted by an FMM model.
`nPeriods`	1	A `"numeric"` value specifying the number of periods at which `vData` is observed.
`timePoints`	`NULL`	A `"numeric"` vector containing the time points per period at which data is observed. When `timePoints = NULL` an equally spaced sequence from 0 to 2π will be assigned.
`nback`	1	A `"numeric"` value specifying the number of FMM components to be fitted.
`betaOmegaRestrictions`	1 : `nback`	An `"integer"` vector of length `nback` indicating which FMM waves are constrained to have equal β and ω parameters.
`maxiter`	`nback`	A `"numeric"` value specifying the maximum number of iterations for the backfitting algorithm.
`stopFunction`	`alwaysFalse`	Function to check the stopping criterion for the backfitting algorithm.
`lengthAlphaGrid`	48	A `"numeric"` value specifying the grid resolution of the parameter α.
`lengthOmegaGrid`	24	A `"numeric"` value specifying the grid resolution of the parameter ω.
`numReps`	3	A `"numeric"` value specifying the number of times (α, ω) parameters are refined.
`showProgress`	`TRUE`	`TRUE` to display a progress indicator on the console.
`showTime`	`FALSE`	`TRUE` to display execution time on the console.
`parallelize`	`FALSE`	`TRUE` to use parallelized procedure to fit a FMM model.
`restrExactSolution`	`FALSE`	`TRUE` to obtain the optimal solution for the restricted fitting.

Table 2: Description of the input arguments of the `fitFMM()` function and their default values.

There are three key issues in the fitting process: the grid search of the pair (α, ω) to solve the estimation problem of a single FMM wave, the backfitting algorithm used for the estimation of the multicomponent models, and the incorporation of restrictions on β and ω parameters. Each of these issues is controlled by several arguments described below.

- **Grid search of the pair** (α, ω). The `lengthAlphaGrid` and `lengthOmegaGrid` arguments are used to set the grid resolution by specifying the number of equally spaced α and ω values. Thus, the objective function will be evaluated a total number of `(lengthAlphaGrid)`×`(lengthOmegaGrid)` times, so when both arguments are large, the computational demand can be high. By reducing the size of the sequences of the α and ω parameters, the algorithm will be computationally more efficient. However, it may fail to obtain an accurate estimation if the grid resolution is too sparse. An implemented option to fine-tune the estimation of the parameters is to repeat the fitting process a `numReps` of times, in such a way that, at each repetition, a new two-dimensional grid of (α, ω) points is created around the previous estimates. In addition, the `parallelize` argument specifies whether a parallel processing implementation is used.
- **Backfitting algorithm**. The argument `maxiter` sets the maximum number of backfitting iterations. Through the argument `stopFunction`, it is possible to set a stopping criterion. Two criteria have been implemented as stop functions in this package. When `stopFunction = alwaysFalse`,

maxiter iterations will be forced. If `stopFunction = R2()`, the algorithm will be stopped when the difference between the explained variability in two consecutive iterations is less than a value pre-specified in the `difMax` argument of `R2()` function.

- **Restrictions**. The argument `betaOmegaRestrictions` sets the equality constraints for the β and ω parameters. For the unrestricted case, `betaOmegaRestrictions = 1:nback`. To add restrictions, `"integer"` vectors of length m can be passed to this argument, so that positions with the same numeric value correspond to FMM waves whose parameters, β and ω, are forced to be equal. Since restricted fitting can be computationally intensive, a two-nested backfitting algorithm can be used for the estimation of ω parameters when the argument `restrExactSolution = FALSE`.

Object of class `"FMM"`

The `fitFMM()` function outputs an `S4` object of class `"FMM"` which contains the slots presented in Table 3.

Slot	Description
`timePoints`	A `"numeric"` vector containing the time points for each data point if one single period is observed.
`data`	A `"numeric"` vector containing the data to be fitted to an FMM model. Data could be collected over multiple periods.
`summarizedData`	A `"numeric"` vector containing the summarized data at each time point across all considered periods.
`nPeriods`	A `"numeric"` value containing the number of periods in `data`.
`fittedValues`	A `"numeric"` vector of the fitted values by the FMM model.
`M`	A `"numeric"` value of the estimated intercept parameter M.
`A`	An m-element `"numeric"` vector of the estimated FMM wave amplitude parameter(s) A.
`alpha`	An m-element `"numeric"` vector of the estimated FMM wave phase translation parameter(s) α.
`beta`	An m-element `"numeric"` vector of the estimated FMM wave skewness parameter(s) β.
`omega`	An m-element `"numeric"` vector of the estimated FMM wave kurtosis parameter(s) ω.
`SSE`	A `"numeric"` value of the residual sum of squares values.
`R2`	An m-element `"numeric"` vector specifying the explained variance by each of the fitted FMM components.
`nIter`	A `"numeric"` value containing the number of iterations of the backfitting algorithm.

Table 3: Summary of the slots of the `S4` object of class `"FMM"` resulting from fitting an FMM model with m components.

The standard methods implemented for the class `"FMM"` include the functions `summary()`, `show()`, `coef()` and `fitted()`. These methods display relevant information of the FMM fitting, and provide the estimated parameters and fitted values. In addition, two more specific functions have been implemented. Through the `extractWaves()` function, the individual contribution of each FMM wave to the fitted values can be extracted. Finally, the location of the peak and trough of each FMM wave, as well as the value of the signal at these time points, can be estimated using the `getFMMPeaks()` function. The required argument of all these methods and functions is an object of the class `"FMM"`. Particularly, `getFMMPeaks()` has an optional argument: `timePointsIn2pi`, that forces the peak and trough locations to be returned into the interval from 0 to 2π when it is `TRUE`.

Plotting FMM models

The **FMM** package includes the function `plotFMM()` to visualize the results of an FMM fit. The arguments of this function are summarized in Table 4.

An object of class `"FMM"` can be plotted in two ways (see Figure 1). The default graphical representation will be a plot on which original data (as points) and the fitted signal (as a line) are plotted together (left panel in Figure 1). The other possible representation is a component plot for displaying each centered FMM wave separately (right panel in Figure 1). Set the boolean argument `components`

Argument	Default value	Description
`objFMM`	no default value	The object of class `"FMM"` to be plotted.
`components`	`FALSE`	`TRUE` to display a plot of components.
`plotAlongPeriods`	`FALSE`	`TRUE` to plot more than 1 period.
`use_ggplot2`	`FALSE`	`TRUE` to plot with **ggplot2** package.
`legendInComponentsPlot`	`TRUE`	`TRUE` to indicate if a legend should be plotted in the component plot.
`textExtra`	empty string	Extra text to be added to the title of the plot.

Table 4: Description of the input arguments of the `plotFMM()` function and their default values.

`= TRUE` to show a component plot. When `legendInComponentsPlot = TRUE`, a legend appears at the bottom of the component plot to indicate the represented waves. The argument `textExtra` allows an extra text to be added to the title of both graphical representations.

As mentioned above, in some cases, data are collected from different periods. All periods can be displayed simultaneously on the default plot using `plotAlongPeriods = TRUE`. For the component plot, this argument is ignored.

The argument `use_ggplot2` provides a choice between building the plot using base R **graphics** or **ggplot2** packages. By default, the **graphics** package is used. When `use_ggplot2 = TRUE`, a more aesthetic and customizable plot is created using the **ggplot2** package.

Simulating data from an FMM model

Data from an FMM model can be easily simulated using the function `generateFMM()` of the package **FMM**. All input arguments of this function are shown in Table 5, along with a short description and their default values.

Argument	Default value	Description
`M`	no default value	Value of the intercept parameter M.
`A`	no default value	Vector of the FMM wave amplitude parameter A.
`alpha`	no default value	Vector of the FMM wave phase translation parameter α.
`beta`	no default value	Vector of the FMM wave skewness parameter β.
`omega`	no default value	Vector of the FMM wave kurtosis parameter ω.
`from`	0	Initial time point of the simulated data.
`to`	2π	Final time point of the simulated data.
`length.out`	100	Desired length of the simulation.
`timePoints`	`seq(from,to,length = length.out)`	Time points at which the data will be simulated.
`plot`	`TRUE`	`TRUE` when the simulated data should be drawn on a plot.
`outvalues`	`TRUE`	`TRUE` when the numerical simulation should be returned.
`sigmaNoise`	0	Standard deviation of the Gaussian noise to be added.

Table 5: Description of the input arguments of the `generateFMM()` function and their default values.

The main arguments of this function are `M`, `A`, `alpha`, `beta` and `omega`, whereby the values of the FMM model parameters are passed to the function. All these arguments are `"numeric"` vectors of length m, except `M`, which has length 1. Longer and smaller vectors will be truncated or replicated as appropriate.

By default, the data will be simulated at a sequence of 100 equally spaced time points from 0 to 2π. The arguments `from`, `to` and `length.out` control such sequences. The sequence can also be manually set using the argument timePoints, in which case from, to and length.out will be ignored.

The user can add a Gaussian noise by argument sigmaNoise. A positive "numeric" value sets the corresponding standard deviation of the Gaussian noise to be added. To create the normally distributed noise, the rnorm() function is used.

The arguments plot and outvalues, both boolean values, determine the output of the generateFMM() function. When outvalues = TRUE, a "list" with input parameters, time points and simulated data is returned. These elements are named input, t and y, respectively. In addition, a scatter plot of y against t can be drawn by setting plot = TRUE.

Basic usage of the FMM package

The example below, based on FMM synthetic data, illustrates the basic uses and capabilities of the functions implemented in the **FMM** package. A set of 100 observations is simulated from an FMM_4 model with intercept parameter $M = 3$, amplitude parameters: $A_1 = 4$, $A_2 = 3$, $A_3 = 1.5$ and $A_4 = 1$, and phase translation parameters: $\alpha_1 = 3.8$, $\alpha_2 = 1.2$, $\alpha_3 = 4.5$ and $\alpha_4 = 2$. With regard to the shape parameters, pairs of waves are equal. Specifically, the shape parameters satisfy:

$$\begin{aligned} \beta_1 = \beta_2 = 3 \qquad\qquad & \omega_1 = \omega_2 = 0.1 \\ \beta_3 = \beta_4 = 1 \qquad\qquad & \omega_3 = \omega_4 = 0.05 \end{aligned}$$

The standard deviation of the error term is set at $\sigma = 0.3$. We use the function generateFMM() to simulate this data set. A set.seed() statement is used to guarantee the reproducibility of the results.

```
> library("FMM")
> set.seed(1115)
> rfmm.data <-generateFMM(M = 3, A = c(4,3,1.5,1), alpha = c(3.8,1.2,4.5,2),
+                         beta = c(rep(3,2),rep(1,2)),
+                         omega = c(rep(0.1,2),rep(0.05,2)),
+                         plot = FALSE, outvalues = TRUE,
+                         sigmaNoise = 0.3)
```

The estimation of an FMM_4 can be performed by setting nback = 4 in the fitting function fitFMM(). The betaOmegaRestrictions parameter allows a wide variety of shape restrictions to be incorporated into the fitting procedure. In this example, to impose the shape restriction on the fitting process, we use betaOmegaRestrictions = c(1,1,2,2).

```
> fit.rfmm <- fitFMM(vData = rfmm.data$y, timePoints = rfmm.data$t, nback = 4,
+                    betaOmegaRestrictions = c(1, 1, 2, 2))
|--------------------------------------------------|
|==================================================|
 Stopped by reaching maximum iterations (4 iteration(s))
```

The results are displayed by the function summary():

```
> summary(fit.rfmm)

Title:
FMM model with 4 components

Coefficients:
M (Intercept): 3.1661
                  A  alpha   beta  omega
FMM wave 1:  4.0447 3.8048 3.0238 0.0930
FMM wave 2:  3.1006 1.1956 3.0238 0.0930
FMM wave 3:  1.6069 4.5228 1.0145 0.0427
FMM wave 4:  1.1194 1.9788 1.0145 0.0427

Peak and trough times and signals:
             t.Upper Z.Upper t.Lower Z.Lower
FMM wave 1:   0.6741  5.3198  4.9354 -2.7565
FMM wave 2:   4.3482  3.4702  2.3263 -2.1742
FMM wave 3:   1.5345 -1.2330  1.3338 -4.1527
FMM wave 4:   5.2737 -1.7005  5.0730 -3.7565
```

```
Residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.719769 -0.162649  0.007025  0.000000  0.160127  0.904218

R-squared:
Wave 1 Wave 2 Wave 3 Wave 4  Total
0.5049 0.3906 0.0531 0.0276 0.9761
```

The FMM wave parameter estimates, as well as the peak and trough times, together with the signal values at those times, are presented in tabular form, where each row corresponds to a component and each column to an FMM wave parameter. As part of the summary, a brief description of the residuals, the proportion of variance explained by each FMM component and by the global model are also shown. The `summary()` output can be assigned to an object to get a `"list"` of all the displayed results.

Other options to return the results are the functions `coef()`, `getFMMPeaks()` and `resid()`. The first two return a `"list"` similar to those obtained with `summary()`. The `resid()` method can be used to obtain the complete residuals vector. In addition, the fitted values can be extracted by the function `fitted()`, which returns a `"data.frame"` with two columns: time points and fitted values.

The FMM plots can be generated in the R **graphics** or **ggplot2** packages. In the code example given below, we use `use_ggplot2 = TRUE` to build Figure 1 based on **ggplot2**. The use of **ggplot2** makes it easier to customize our plots and modify features, such as scales, margins, axes, etc. In Figure 1, the two possible FMM plots are arranged via the `grid.arrange()` function of the **gridExtra** package (Auguie, 2017).

```
> library("RColorBrewer")
> library("ggplot2")
> library("gridExtra")
> # Plot the fitted FMM model
> titleText <- "Simulation of four restricted FMM waves"
> defaultrFMM2 <- plotFMM(fit.rfmm, use_ggplot2 = TRUE, textExtra = titleText) +
+                 theme(plot.margin=unit(c(1,0.25,1.3,1), "cm")) +
+                 ylim(-5, 6)
> comprFMM2 <- plotFMM(fit.rfmm, components=TRUE, use_ggplot2 = TRUE,
+                      textExtra = titleText) +
+              theme(plot.margin=unit(c(1,0.25,0,1), "cm")) +
+              ylim(-5, 6) +
+              scale_color_manual(values = brewer.pal("Set1",n = 8)[3:6])
> grid.arrange(defaultrFMM2, comprFMM2, nrow = 1)
```

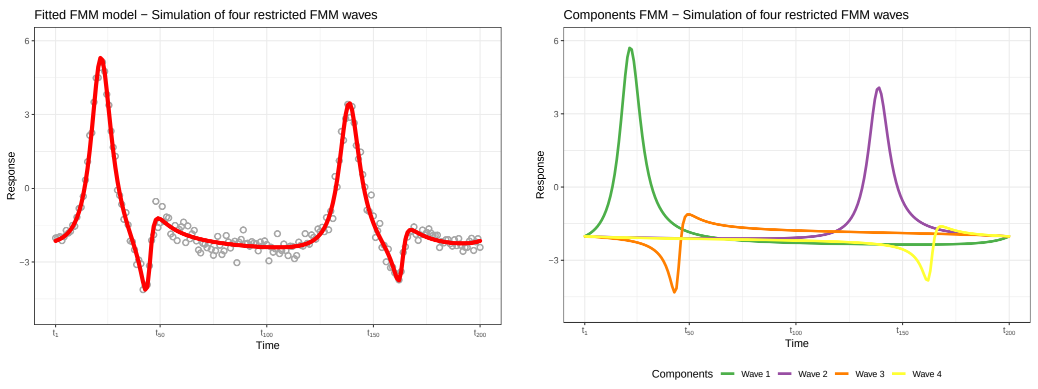

Figure 1: Graphical representation of the estimated restricted FMM_4 signal with $\beta_1 = \beta_2, \omega_1 = \omega_2$ and $\beta_3 = \beta_4, \omega_3 = \omega_4$ constraints. A scatter plot of the simulated data along with the fitted signal is displayed on the left (default plot). The component plot is shown on the right.

Real data analysis using the FMM package

This section illustrates the use of the **FMM** package on the analysis of real signals from chronobiology, electrocardiography and neuroscience. To do this, the package includes four real-world data sets in `RData` format which are described in the following sections.

Example 1: Chronobiology

Chronobiology studies ubiquitous daily variations found in nature and in many aspects of the physiology of human beings, such as blood pressure or hormone levels (Mermet et al., 2017). These phenomena commonly display signals with oscillatory patterns that repeat every 24 hours, usually known as circadian rhythms. In particular, circadian gene expression data have been deeply analyzed in the literature as they regulate the vast majority of molecular rhythms involved in diverse biochemical and cellular functions, see among others Zhang et al. (2014), Cornelissen (2014) and Larriba et al. (2020).

The **FMM** package includes a data set called mouseGeneExp that contains expression data of the *Iqgap2* gene from mouse liver. The liver circadian database is widely extended in chronobiology since the liver is a highly rhythmic organ with moderate levels of noise (Anafi et al., 2017; Larriba et al., 2018, 2020). The complete database is freely available at NCBI GEO (http://www.ncbi.nlm.nih.gov/geo/), with GEO accession number GSE11923. Gene expression values are given along 48 hours with a sampling frequency of 1 hour/2 days. Hence, data are collected along two periods, and an FMM$_1$ model is fitted to the *Iqgap2* average expressed values as follows:

```
> data("mouseGeneExp", package = "FMM")
> fitGene <- fitFMM(vData = mouseGeneExp, nPeriods = 2, nback = 1, showProgress = FALSE)
> summary(fitGene)

Title:
FMM model with 1 components

Coefficients:
M (Intercept): 10.1508
                  A  alpha   beta  omega
FMM wave 1:  0.4683 3.0839 1.5329 0.0816

Peak and trough times and signals:
             t.Upper Z.Upper t.Lower Z.Lower
FMM wave 1:   0.1115 10.6191  6.0686  9.6825

Residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-9.751e-02 -3.490e-02  2.269e-03 -1.530e-06  2.670e-02  1.890e-01

R-squared:
[1] 0.8752
```

The behavior of the FMM versus COS model to describe this asymmetric pattern has been compared in terms of R^2. The FMM model clearly outperforms the COS one with an R^2 of 0.8752 and 0.2835, respectively. In addition, a difference of 4.73 hours in peak time estimation between both models is observed, the FMM peak estimate being much more reliable, as is shown in Figure 2.

Example 2: Electrocardiography

ECG records the periodic electrical activity of the heart. This activity represents the contraction and relaxation of the atria and ventricle, processes related to the crests and troughs of the ECG waveform. Heartbeats are decomposed into five fundamental waves, labelled as P, Q, R, S and T, corresponding to the different phases of the heart's electric activity. The main features used in medical practice for cardiovascular pathology diagnosis are related to the location and amplitudes of these waves, and, of them, those labeled as P, R and T are of particular interest (Bayes de Luna, 2007). Standard ECG signals are registered using twelve leads, calculated from different electrode locations. Lead II is the reference signal, as it usually provides a good view of the main ECG waves (Meek and Morris, 2002).

The **FMM** package includes the analysis of a typical ECG heartbeat from the QT database (Laguna et al., 1997). This recording, from the subject *sel100*, belongs to the *Normal* category, regarding Physionet's pathology classification (Goldberger et al., 2000). The data illustrate the voltage of the heart's electric activity, measured in mV, along the heartbeat with a sampling frequency of 250Hz. Specifically, the ECG signal from lead II in the fifth of the thirty annotated heartbeats is analysed. Recordings are publicly available on (http://www.physionet.org). Data are saved as ecgData in the package. For an ECG heartbeat, an FMM$_{ecg}$, a fifth order multicomponent FMM model can be fitted with the instruction:

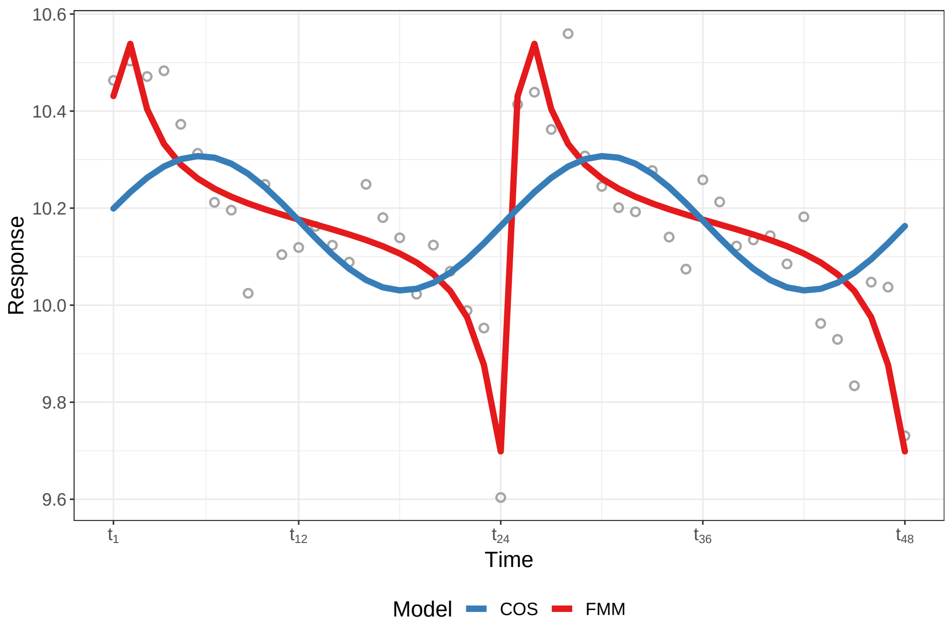

Figure 2: *Iqgap2* gene expression data along two periods (grey dots); FMM (red line) and COS (blue line) fitted signals.

```
> data("ecgData", package = "FMM")
> fitEcg <- fitFMM(ecgData, nback = 5, showProgress = FALSE)
> summary(fitEcg)

Title:
FMM model with 5 components

Coefficients:
M (Intercept): 5.2717
                  A  alpha   beta  omega
FMM wave 1:  0.6454 5.5151 3.2926 0.0325
FMM wave 2:  0.0994 4.4203 3.7702 0.1356
FMM wave 3:  0.2443 5.3511 0.6636 0.0323
FMM wave 4:  0.3157 5.5919 4.8651 0.0126
FMM wave 5:  0.0666 1.7988 2.1277 0.1632

Peak and trough times and signals:
             t.Upper Z.Upper t.Lower Z.Lower
FMM wave 1:   2.3686  6.2370  3.1841  4.7241
FMM wave 2:   1.1905  4.9487  2.0693  4.6897
FMM wave 3:   2.3965  6.0828  2.1872  4.5551
FMM wave 4:   2.4210  5.7933  2.4719  4.7175
FMM wave 5:   5.1212  4.8646  4.3689  4.7228

Residuals:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.0690885 -0.0095597 -0.0001127  0.0000000  0.0098533  0.0623569

R-squared:
Wave 1 Wave 2 Wave 3 Wave 4 Wave 5  Total
0.7645 0.0920 0.0581 0.0493 0.0278 0.9918
```

It is worth noting that the **FMM** package not only provides ECG signal-fitting (the left hand panel in Figure 3), but it also does wave decomposition and fiducial mark annotations on the desired waves (the right hand panel in Figure 3). It is clearly visible how the specific shapes of the five main waves contribute to drawing and explaining the lead II ECG waveform from the *Normal* morphology. See Rueda et al. (2021b) for a complete review of FMM$_{ecg}$.

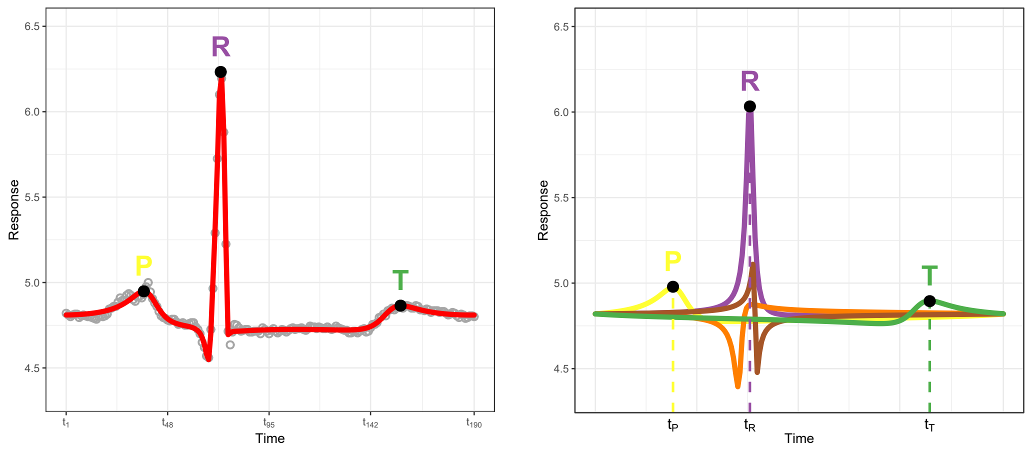

Figure 3: FMM$_{ecg}$ performance on a single beat from patient *sel100* from the QT database. Left: Data (grey dots) and FMM fitting (red line). Black dots locate the P, R and T fiducial marks. Right: ECG decomposition on P(orange), Q (purple), R (green), S (yellow) and T (blue) waves. Dash lines indicate P, R and T peak times.

Example 3: Neuroscience

Single AP curve

The study of the electrophysiological activity of neurons is one of the main research branches in neuroscience. The AP curves are oscillatory signals that serve as basic information units between neurons. They measure the electrical potential difference between inside and outside the cell due to an external stimulus. Gerstner et al. (2014) can serve as a basic reference for electrophysiological neuroscience. Recently, the shape and other features of the AP have been used in problems such as spike sorting (Rácz et al., 2020; Souza et al., 2019; Caro-Martín et al., 2018) or neuronal cell type classification (Teeter et al., 2018; Gouwens et al., 2019; Mosher et al., 2020; Rodríguez-Collado and Rueda, 2021b).

The package includes an example of a neuronal AP. The data were simulated with the renowned Hodgkin-Huxley model, first presented in Hodgkin and Huxley (1952), which is defined as a system of ordinary differential equations and has been used in a wide array of applications, as it successfully describes the neuronal activity in various organisms. The simulation has been done using a modified version of the python package NeuroDynex available at Gerstner et al. (2014). More concretely, a short square stimulus of $12\mu A$ has been applied to the neuron. The data can be accurately fitted by an FMM$_2$ model as follows:

```
> data("neuronalSpike", package = "FMM")
> fitSingleAP <- fitFMM(neuronalSpike, nback = 2, showProgress = FALSE)
> summary(fitSingleAP)

Title:
FMM model with 2 components

Coefficients:
M (Intercept): 44.9474
                    A  alpha   beta  omega
FMM wave 1:  52.9014 4.4160 3.0606 0.0413
FMM wave 2:  18.5046 4.6564 4.9621 0.0322

Peak and trough times and signals:
             t.Upper  Z.Upper t.Lower  Z.Lower
FMM wave 1:   1.2777 110.8361  5.9669  -2.5002
FMM wave 2:   1.4319  36.9084  1.5649 -16.2572

Residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
```

```
-14.3012  -1.0038   0.7472   0.0000   1.3230  24.8618

R-squared:
Wave 1 Wave 2  Total
0.9064 0.0604 0.9669
```

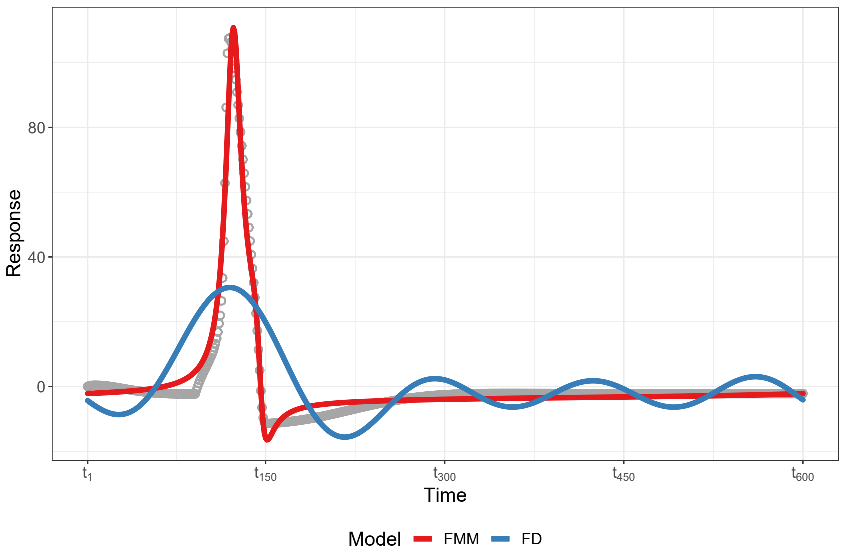

Figure 4: Neuronal AP simulated with the Hodgkin-Huxley model (parameters: $C = 1, g_{Na} = 260, g_K = 30, g_L = 0.31, V_K = -12, V_{Na} = 115, V_L = 10.6, \widetilde{a}_n = 1.15, \widetilde{b}_n = 0.85, \widetilde{a}_m = 0.9, \widetilde{b}_m = 1.3, \widetilde{a}_h = 1, \widetilde{b}_h = 1$ and applying a current of $12\mu A$ for 1 millisecond) and the estimated FMM$_2$ signal in red. An FD model of the same number of degree of freedom has been fitted and plotted in blue.

The goodness of fit of the FMM$_2$ model can be ascertained in Figure 4. For comparison purposes, an FD model has been fitted with the same number of degrees of freedom. While the FD attains an $R^2 = 0.3926$, the FMM model achieves a better fit with $R^2 = 0.9669$.

AP train

Multiple AP curves, denominated spike or AP train, are usually observed as the response to a stimulus. Various models, such as the widely used leaky-and-fire models (Lynch and Houghton, 2015), cut the signal into segments, each one containing an AP curve. Some authors suggest cutting the signal into even segments (Gerstner et al., 2014). However, the length of the segments turns out to be significantly different between different types of neurons, as explained in Teeter et al. (2018), and unequal data segments can lessen the utility of some approaches. An important aspect to take into account is that the shape of the APs in the spike train is considered to be similar and, consequently, a restricted FMM model can accurately fit the entire signal.

The **FMM** package includes the data of a spike train composed of three AP curves. The proposed model for use with these data is an FMM$_{ST}$ model, as defined in Rodríguez-Collado and Rueda (2021a). Each AP is modeled by two components. The β and ω parameters are constrained between AP curves. The code below fits the model.

```
> data("neuronalAPTrain", package = "FMM")
> nAPs <- 3; restriction <- c(rep(1,nAPs),rep(2,nAPs))
> fitAPTrain<-fitFMM(neuronalAPTrain, nback = nAPs*2,
                     betaRestrictions = restriction,
                     omegaRestrictions = restriction,
                     showProgress = FALSE, parallelize=TRUE)
> summary(fitAPTrain)

Title:
FMM model with 6 components

Coefficients:
M (Intercept): 135.4137
```

```
                    A  alpha   beta  omega
FMM wave 1:  51.7069 6.1358 2.8172 0.0384
FMM wave 2:  52.0915 1.7541 2.8172 0.0384
FMM wave 3:  51.1140 4.2319 2.8172 0.0384
FMM wave 4:  20.3725 4.4778 4.8637 0.0552
FMM wave 5:  19.2429 1.9981 4.8637 0.0552
FMM wave 6:  19.6748 0.0973 4.8637 0.0552

Peak and trough times and signals:
             t.Upper  Z.Upper t.Lower  Z.Lower
FMM wave 1:   3.0067 111.4319  2.5332  -1.2051
FMM wave 2:   4.9082 111.7323  4.4347  -1.4607
FMM wave 3:   1.1028 111.2700  0.6293  -0.1561
FMM wave 4:   1.2077  58.5986  1.4310 -14.2508
FMM wave 5:   5.0113  58.5537  5.2345 -13.3010
FMM wave 6:   3.1104  58.4889  3.3337 -13.7041

Residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-14.8618  -1.4929   0.5029   0.0000   1.6021  19.1978

R-squared:
Wave 1 Wave 2 Wave 3 Wave 4 Wave 5 Wave 6  Total
0.2524 0.2881 0.3501 0.0244 0.0276 0.0413 0.9839
```

In Figure 5, the fit of the FMM$_{ST}$ model can be visualized. The goodness of fit of the model is excellent, achieving an $R^2 = 0.9839$.

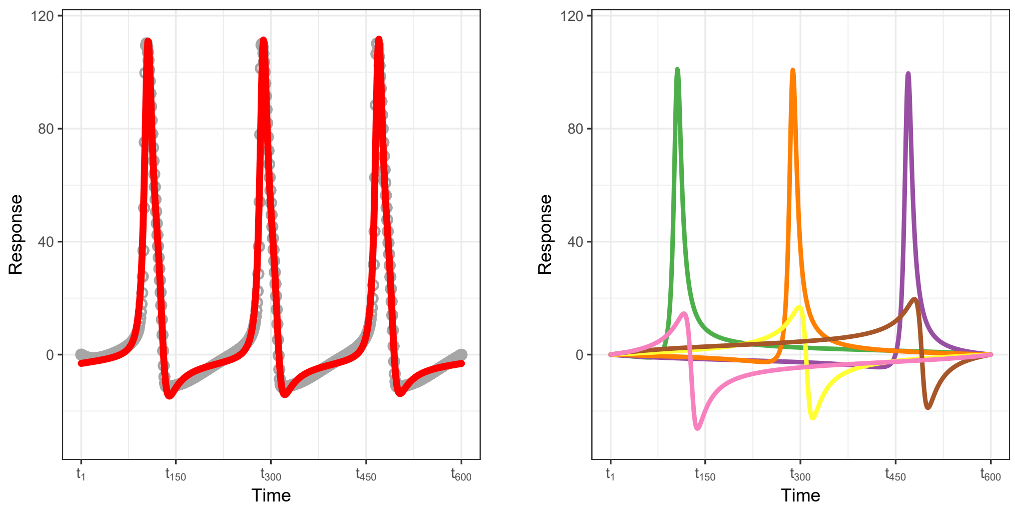

Figure 5: Neuronal APs simulated with the Hodgkin-Huxley model (parameters: $C = 1, g_{Na} = 232, g_K = 45, g_L = 0.215, V_K = -12, V_{Na} = 115, V_L = 10.6, \widetilde{a}_n = 0.95, \widetilde{b}_n = 1.3, \widetilde{a}_m = 1, \widetilde{b}_m = 1.15, \widetilde{a}_h = 1, \widetilde{b}_h = 1$ and applying a short square current of 4.5 μA for 1 millisecond) and the estimated FMM$_{ST}$ signal in red. The components plot of the model can be seen on the right hand side of the figure.

Summary

A general overview on the R package **FMM**, which implements the estimation of FMM models, is provided in this paper. The flexibility offered by these models to fit oscillatory signals of many different shapes makes them a very useful tool to model complex rhythmic patterns. The FMM methodology and its application to very diverse biological data has been described in previous papers (Rueda et al., 2019, 2021b,c) and recently revised in Rueda et al. (2021a).

The package allows both single and multicomponent FMM models to be estimated. In order to provide greater flexibility, equality constraints for shape parameters have also been implemented. In addition, graphical representations of the fitted models and the possibility of generating synthetic data are available. The functionality of the package has been illustrated by simulated data and also by real examples from different areas of application related to present-day biological problems. The latest release of the **FMM** package is publicly available on CRAN (http://CRAN.R-project.org/package=FMM). A development version is also provided via GitHub at https://github.com/alexARC26/FMM where code contributions and bugs can be reported.

Possible future extensions of the **FMM** package include the implementation of additional restrictions to suit the model to other real signals; the possibility to include weights that determine how much each observation influences the parameter estimates; and the choice of an optimization technique, other than the Neldel-Mead method, in the estimation algorithm.

Acknowledgments

This research was partially supported by the Spanish Ministerio de Economía y Competitividad, grant PID2019-106363RB-I00, as well as the Call for predoctoral contracts of the UVa 2020. The authors thank the editors and two anonymous reviewers for their constructive comments that helped to improve this paper and the described package.

Bibliography

R. C. Anafi, L. J. Francey, J. B. Hogenesch, and J. Kim. CYCLOPS reveals human transcriptional rhythms in health and disease. *Proceedings of the National Academy of Sciences*, 114(20):5312–5317, 2017. URL https://doi.org/10.1073/pnas.1619320114. [p353]

B. Auguie. *gridExtra: Miscellaneous Functions for Grid Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3. [p352]

A. Bayes de Luna. *Basic Electrocardiography: Normal and Abnormal ECG Patterns*. John Wiley & Sons, Ltd, 2007. URL https://doi.org/10.1002/9780470692622. [p353]

B. Boashash. *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. Academic Press, San Francisco, CA, 2nd edition, 2016. [p343]

M. Carlucci, A. Kriščiūnas, H. Li, P. Gibas, K. Koncevičius, A. Petronis, and G. Oh. DiscoRhythm: An easy-to-use web application and R package for discovering rhythmicity. *Bioinformatics*, 36(6): 1952–1954, 2020. URL https://doi.org/10.1093/bioinformatics/btz834. [p343]

Caro-Martín, Delgado-García, Gruart, and Sánchez-Campusano. Spike sorting based on shape, phase, and distribution features, and K-TOPS clustering with validity and error indices. *Scientific Reports*, 8 (1):1–28, 2018. URL https://doi.org/10.1038/s41598-018-35491-4. [p355]

W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2021. URL https://CRAN.R-project.org/package=shiny. R package version 1.6.0. [p343]

G. Cornelissen. Cosinor-based rhythmometry. *Theoretical Biology and Medical Modelling*, 11:16, 2014. URL https://doi.org/10.1186/1742-4682-11-16. [p343, 353]

M. Corporation and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2020. URL https://CRAN.R-project.org/package=doParallel. R package version 1.0.16. [p347]

T. D. Downs and K. V. Mardia. Circular regression. *Biometrika*, 89(3):683–698, 2002. URL https://doi.org/10.1093/biomet/89.3.683. [p344]

I. Fernández, A. Rodríguez-Collado, Y. Larriba, A. Lamela, C. Canedo, and C. Rueda. *FMM: Rhythmic Patterns Modeling by FMM Models*, 2021. URL https://CRAN.R-project.org/package=FMM. R package version 0.3.0. [p344]

W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. URL https://doi.org/10.1017/CBO9781107447615. [p355, 356]

C. L. Gierke, R. Helget, and G. Cornelissen-Guillaume. *CATkit: Chronomics Analysis Toolkit (CAT): Periodicity Analysis*, 2018. URL https://CRAN.R-project.org/package=CATkit. R package version 3.3.3. [p343]

A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):E215–20, 2000. URL https://doi.org/10.1161/01.cir.101.23.e215. [p353]

N. W. Gouwens, S. A. Sorense, J. Berg, C. Lee, et al. Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature Neuroscience*, 22:1182–1195, 2019. URL https://doi.org/10.1038/s41593-019-0417-0. [p355]

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952. URL https://doi.org/10.1113/jphysiol.1952.sp004764. [p355]

M. E. Hughes, J. B. Hogenesch, and K. Kornacker. JTK_CYCLE: An efficient nonparametric algorithm for detecting rhythmic components in genome scale data sets. *Journal of Biological Rhythms*, 25(5): 372–380, 2010. URL https://doi.org/10.1177/0748730410379711. [p343]

S. Kato, K. Shimizu, and G. S. Shieh. A circular - circular regression model. *Statistica Sinica*, 18(2): 633–645, 2008. URL https://www.jstor.org/stable/24308499. [p344]

P. Laguna, R. G. Mark, A. Goldberg, and G. B. Moody. A database for evaluation of algorithms for measurement of qt and other waveform intervals in the ecg. In *Computers in Cardiology 1997*, pages 673–676. IEEE, 1997. URL https://doi.org/10.1109/CIC.1997.648140. [p353]

Y. Larriba, C. Rueda, M. A. Fernández, and S. D. Peddada. A bootstrap based measure robust to the choice of normalization methods for detecting rhythmic features in high dimensional data. *Frontiers in Genetics*, 9:24, 2018. URL https://doi.org/10.3389/fgene.2018.00024. [p353]

Y. Larriba, C. Rueda, M. A. Fernández, and S. D. Peddada. Order restricted inference in chronobiology. *Statistics in Medicine*, 39(3):265–278, 2020. URL https://doi.org/10.1002/sim.8397. [p353]

E. P. Lynch and C. J. Houghton. Parameter estimation of neuron models using in-vitro and in-vivo electrophysiological data. *Frontiers in Neuroinformatics*, 9:10, 2015. URL https://doi.org/10.3389/fninf.2015.00010. [p356]

S. Meek and F. Morris. Introduction. I—Leads, rate, rhythm, and cardiac axis. *BMJ*, 324(7334):415–418, 2002. ISSN 0959-8138. doi: 10.1136/bmj.324.7334.415. [p353]

J. Mermet, J. Yeung, and F. Naef. Systems chronobiology: Global analysis of gene regulation in a 24-hour periodic world. *Cold Spring Harbor Perspectives in Biology*, 9(3), 2017. URL https://doi.org/10.1101/cshperspect.a028720. [p353]

C. P. Mosher, Y. Wei, J. Kamiński, A. Nandi, A. N. Mamelak, C. A. Anastassiou, and U. Rutishauser. Cellular classes in the human brain revealed in vivo by heartbeat-related modulation of the extracellular action potential waveform. *Cell Reports*, 30(10):3536–3551.e6, 2020. URL https://doi.org/10.1016/j.celrep.2020.02.027. [p355]

M. Moskon. CosinorPy: A python package for cosinor-based rhythmometry. *BMC Bioinformatics*, 21 (1):485, 2020. URL https://doi.org/10.1186/s12859-020-03830-w. [p343]

A. Mutak. *cosinor2: Extended Tools for Cosinor Analysis of Rhythms*, 2018. URL https://CRAN.R-project.org/package=cosinor2. R package version 0.2.1. [p343]

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4): 308–313, 1965. URL https://doi.org/10.1093/comjnl/7.4.308. [p345]

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL https://CRAN.R-project.org/package=RColorBrewer. R package version 1.1-2. [p347]

R. Parsons, R. Parsons, N. Garner, H. Oster, and O. Rawashdeh. CircaCompare: A method to estimate and statistically support differences in mesor, amplitude and phase, between circadian rhythms. *Bioinformatics*, 36(4):1208–1212, 2020. URL https://doi.org/10.1093/bioinformatics/btz730. [p343]

M. Rácz, C. Liber, E. Németh, R. Fiáth, J. Rokai, I. Harmati, I. Ulbert, and G. Márton. Spike detection and sorting with deep learning. *Journal of Neural Engineering*, 17(1):016038, 2020. URL https://doi.org/10.1088/1741-2552/ab4896. [p355]

W. Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2021. URL https://CRAN.R-project.org/package=psych. R package version 2.1.6. [p343]

A. Rodríguez-Collado and C. Rueda. A simple parametric representation of the Hodgkin-Huxley model. *PLoS ONE*, 16(7):e0254152, 2021a. URL https://doi.org/110.1371/journal.pone.0254152. [p344, 346, 356]

A. Rodríguez-Collado and C. Rueda. Electrophysiological and transcriptomic features reveal a circular taxonomy of cortical neurons. *Frontiers in Human Neuroscience*, 15:410, 2021b. URL https://doi.org/10.3389/fnhum.2021.684950. [p355]

C. Rueda, Y. Larriba, and S. D. Peddada. Frequency modulated Möbius model accurately predicts rhythmic signals in biological and physical sciences. *Scientific Reports*, 9(1):18701, 2019. URL https://doi.org/10.1038/s41598-019-54569-1. [p344, 357]

C. Rueda, I. Fernández, Y. Larriba, and A. Rodríguez-Collado. The FMM approach to analyze biomedical signals: Theory, software, applications and future. *Mathematics*, 9(10):1145, 2021a. URL https://doi.org/10.3390/math9101145. [p357]

C. Rueda, Y. Larriba, and A. Lamela. The hidden waves in the ECG uncovered revealing a sound automated interpretation method. *Scientific Reports*, 11:3724, 2021b. URL https://doi.org/10.1038/s41598-021-82520-w. [p344, 354, 357]

C. Rueda, A. Rodríguez-Collado, and Y. Larriba. A novel wave decomposition for oscillatory signals. *IEEE Transactionns on Signal Processing*, 69:960–972, 2021c. URL https://doi.org/10.1109/TSP.2021.3051428. [p344, 357]

M. Sachs. *cosinor: Tools for estimating and predicting the cosinor model*, 2014. URL https://CRAN.R-project.org/package=cosinor. R package version 1.1. [p343]

A. S. Shah. *card: Cardiovascular and Autonomic Research Design*, 2020. URL https://CRAN.R-project.org/package=card. R package version 0.1.0. [p343]

J. M. Singer and J. J. Hughey. LimoRhyde: A flexible approach for differential analysis of rhythmic transcriptome data. *Journal of Biological Rhythms*, 34(1):5–18, 2019. URL https://doi.org/10.1177/0748730418813785. [p343]

B. C. Souza, V. L. dos Santos, J. Bacelo, and A. B. Tort. Spike sorting with Gaussian mixture models. *Scientific Reports*, 9(1):1–14, 2019. URL https://doi.org/10.1038/s41598-019-39986-6. [p355]

C. Teeter, R. Iyer, V. Menon, N. Gouwens, D. Feng, J. Berg, A. Szafer, N. Cain, H. Zeng, M. Hawrylycz, C. Koch, and S. Mihalas. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature Communications*, 9(1):1–15, 2018. URL https://doi.org/10.1038/s41467-017-02717-4. [p355, 356]

P. F. Thaben and P. O. Westermark. Detecting rhythms in time series with RAIN. *Journal of Biological Rhythms*, 29(6):391–400, 2014. URL https://doi.org/10.1177/0748730414553029. [p343]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p347]

R. Zhang, N. F. Lahens, H. I. Ballance, M. E. Hughes, and J. B. Hogenesch. A circadian gene expression atlas in mammals: Implications for biology and medicine. *Proceedings of the National Academy of Sciences*, 111(45):16219–16224, 2014. URL https://doi.org/10.1073/pnas.1408886111. [p353]

Itziar Fernández
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0002-5077-4448
itziar.fernandez@uva.es

Alejandro Rodríguez-Collado
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0001-5450-9580
alejandro.rodriguez.collado@uva.es

Yolanda Larriba
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0003-0254-4928
yolanda.larriba@uva.es

Adrián Lamela
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0002-7155-8832
adrianlamela@gmail.com

Christian Canedo
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0001-6731-7369
christian.canedo@alumnos.uva.es

Cristina Rueda
Department of Statistics and Operations Research
Universidad de Valladolid
Valladolid, Spain
ORCiD: 0000-0001-9638-8991
cristina.rueda@uva.es