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

**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 neuronal activity.

**Keywords** oscillation · rhythmic signal · frequency modulated Möbius model · FMM · restricted FMM model · R

1 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, ...]
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 [R Core Team 2020], 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 2018], dedicated to personality and psychological research; or the function cosinor contained in a very 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 non-sinusoidal, 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 over-fitting 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 (https://bioconductor.org/packages/DiscoRhythm) with a web interface based on the R Shiny platform [Chang et al. 2020]. 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. 2019], an R package implemented for modeling cosinoidal curves by nonlinear regression and available from GitHub (https://github.com/RWParsons/circacompare). Also hosted on GitHub, we can 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 non-sinusoidal 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 2021b]. 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, 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: Section 2 provides a brief overview of both mono and multicomponent FMM models; Section 3 is dedicated to the implementation details of the FMM package. Through simulated examples, Section 4 introduces the basic usage of the package, including the data generation and the fitting, as well as the visualization of the results. In Section 5 the FMM package performance is shown through three application areas governed by oscillatory systems: chronobiology, ECG and neuroscience. Finally, a summary is provided in Section 6.

2 Frequency modulated Möbius (FMM) model

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 $W(t; \nu) = A \cos(\phi(t; \alpha, \beta, \omega))$ where $\nu = (A, \alpha, \beta, \omega)^T$, $A \in \mathbb{R}^+$ represents the wave amplitude and,

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

the wave phase. The phase angle $\phi$ 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 non-sinusoidal 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 $\alpha$, $\beta$, and $\omega$ 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 $\beta$ close to 0, $\pi$ or $2\pi$. When $\beta = \frac{\pi}{2}$ or $\beta = \frac{3\pi}{2}$, extreme asymmetric patterns are described. Moreover, a value of $\omega$ close to zero describes an extreme spiked wave and, as $\omega$ 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,

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

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

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

### 2.1 Monocomponent FMM model

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

$$X(t_i) = M + W(t_i; \nu) + e(t_i), \quad i = 1, \ldots, n \quad (3)$$

where $M \in \mathbb{R}$ is an intercept parameter describing the baseline level of the signal, $W(t_i; \nu)$ 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 $\sigma^2$.

#### 2.1.1 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 $(\alpha, \omega)$ parameters is performed. For each pair of $(\alpha, \omega)$ fixed values, the estimates for $\hat{M}$, $A$ and $\beta$ are obtained by solving a least square problem as detailed below.

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

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

where $t_i^* = \alpha + 2 \arctan(\omega \tan(\frac{\beta - \alpha}{2}))$, $\varphi = \beta - \alpha$, 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:

$$X(t_i) = M + \delta z_i + \gamma w_i + e(t_i) \quad (5)$$

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

Since $\alpha$ and $\omega$ are fixed, the estimates for $M$, $\delta$ and $\gamma$ are obtained by minimizing the residual sum of the squares (RSS),

$$RSS = \sum_{i=1}^n (X(t_i) - (\hat{M} + \hat{\delta} z_i + \hat{\gamma} w_i))^2 \quad (6)$$
And the estimates for $M$, $A$ and $\beta$ are straightforward to derive as follows,

\[
\hat{M} = \bar{X} - \delta \sum_{i=1}^{n} z_i - \hat{\gamma} \sum_{i=1}^{n} w_i
\]

\[\begin{align*}
\hat{\alpha} & = \sqrt{\hat{\delta}^2 + \hat{\gamma}^2} \\
\hat{\beta} & = \alpha + \varphi
\end{align*}\]

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:

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (X(t_i) - \hat{X}(t_i))^2}{\sum_{i=1}^{n} (X(t_i) - \bar{X})^2}
\]

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

2.2 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:

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (X(t_i) - \hat{X}(t_i))^2}{\sum_{i=1}^{n} (X(t_i) - \bar{X})^2}
\]

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

2.2.2 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 $J$th FMM wave at $t_i$, $i = 1, \ldots, n$ in the $k$th iteration. The algorithm is structured as follows:

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

\[
r_J^{(k)}(t_i) = X(t_i) - \sum_{j=1}^{J-1} \hat{W}_J^{(k)}(t_i) - \sum_{j=J+1}^{m} \hat{W}_J^{(k-1)}(t_i)
\]

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: $R^2_k - R^2_{k-1} \leq C$, where $R^2_k$ (defined in Equation 11) is the proportion of variance explained by the model in the $k$th iteration and $C$ a constant.
4. $\hat{M}$ and $\hat{A}_J$ are derived by solving

\[
\min_{M \in \mathbb{R}, A_J \in \mathbb{R}^n} \sum_{i=1}^{n} (X(t_i) - M - \sum_{J=1}^{m} A_J \cos(\hat{\phi}_J(t_i)))^2
\]

where $\hat{\phi}_J(t_i) = \phi(t_i; \hat{\alpha}_J, \hat{\beta}_J, \hat{\omega}_J)$ defined in Equation 11.
2.3 Restricted multicomponent FMM model

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

\[
\beta_1 = \ldots = \beta_{m_1} \\
\beta_{m_1+1} = \ldots = \beta_{m_2} \\
\vdots \\
\beta_{m_d-1+1} = \ldots = \beta_{m_d}
\]

and/or $d'$ blocks of restrictions on $\omega$ parameters:

\[
\omega_1 = \ldots = \omega_{m'_1} \\
\omega_{m'_1+1} = \ldots = \omega_{m'_2} \\
\vdots \\
\omega_{m'_d'-1+1} = \ldots = \omega_{m'_d'}
\]

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

2.3.1 FMM estimation algorithm with restrictions on the $\beta$ 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:

\[
\hat{\beta}^*_J = \text{angularMean}(\hat{\beta}_1, \ldots, \hat{\beta}_{m_1}) \quad J = 1, \ldots, m_1 \\
\hat{\beta}^*_J = \text{angularMean}(\hat{\beta}_{m_1+1}, \ldots, \hat{\beta}_{m_2}) \quad J = m_1 + 1, \ldots, m_2 \\
\vdots \\
\hat{\beta}^*_J = \text{angularMean}(\hat{\beta}_{m_d-1+1}, \ldots, \hat{\beta}_{m_d}) \quad J = m_d-1 + 1, \ldots, m_d
\]

Then, the algorithm continues to the next step.

2.3.2 FMM estimation algorithm with restrictions on the $\omega$ parameters

A block is defined as the set of FMM waves with common $\omega$ parameter. When constraints for the $\omega$ parameters are incorporated, the grid search for the different $\omega$ 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.

3 FMM package: Implementation details

The FMM code makes use of the doParallel package [Corporation and Weston, 2020] to embed parallelization for the restricted version of 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 from Sections 3.1 to 3.4: 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.

3.1 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.
FMM: Frequency Modulated Möbius Models in R

Function Description

**Fitting function**

fitFMM(vData, timePoints, nback, ...) Estimates an \( \mathbf{FMM}_{n\text{back}} \) model to \( \mathbf{vData} \) 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.

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.

The fitting function \( \text{fitFMM()} \) requires the \( \mathbf{vData} \) input argument, which contains the data to be fitted. Two other arguments can be used to control a basic fitting: \( \mathbf{timePoints} \), which contains the specific time points of the single period; and \( \mathbf{nback} \), 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 \( \text{fitFMM()} \) function through the input argument \( \mathbf{nPeriods} \). When \( \mathbf{nPeriods} > 1 \), the FMM fitting is carried out by averaging the data collected at each time point across all considered periods.

| Argument      | Default value | Description                                      |
|---------------|---------------|--------------------------------------------------|
| \( \mathbf{vData} \) | no default value | Data to be fitted by an FMM model. |
| \( \mathbf{nPeriods} \) | 1             | Number of periods at which \( \mathbf{vData} \) is observed. |
| \( \mathbf{timePoints} \) | NULL          | Vector of time points per period at which data is observed. When \( \mathbf{timePoints} = \text{NULL} \), an equally spaced sequence from 0 to \( 2\pi \) will be assigned. |
| \( \mathbf{nback} \) | 1             | Number of FMM components to be fitted. |
| \( \mathbf{betaRestrictions} \) | 1 : \( \mathbf{nback} \) | \( \beta \) constraint vector. |
| \( \mathbf{omegaRestrictions} \) | 1 : \( \mathbf{nback} \) | \( \omega \) constraint vector. |
| \( \mathbf{maxiter} \) | \( \mathbf{nback} \) | Maximum number of iterations for the backfitting algorithm. |
| \( \mathbf{stopFunction} \) | alwaysFalse | Function to check the criterion convergence for the backfitting algorithm. |
| \( \mathbf{lengthAlphaGrid} \) | 48            | Grid resolution of the parameter \( \alpha \). |
| \( \mathbf{lengthOmegaGrid} \) | 24            | Grid resolution of the parameter \( \omega \). |
| \( \mathbf{numReps} \) | 3             | Number of times \((\alpha, \omega)\) parameters are refined. |
| \( \mathbf{showProgress} \) | TRUE         | TRUE to display a progress indicator on the console. |
| \( \mathbf{showTime} \) | TRUE         | TRUE to display execution time on the console. |
| \( \mathbf{parallelize} \) | FALSE        | TRUE to use parallelized procedure to fit restricted \( \mathbf{FMM_{m}} \) model. |

Table 2: Description of the input arguments of the \( \text{fitFMM()} \) function and their default values.

There are three key issues in the fitting process: the grid search of the pair \((\alpha, \omega)\) 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 \( \beta \) and \( \omega \) parameters. Each of these issues is controlled by several arguments described below.

- **Grid search of the pair \((\alpha, \omega)\)**. The \( \mathbf{lengthAlphaGrid} \) and \( \mathbf{lengthOmegaGrid} \) arguments are used to set the grid resolution by specifying the number of equally spaced \( \alpha \) and \( \omega \) values. Thus, the objective function will be evaluated a total number of \((\mathbf{lengthAlphaGrid}) \times (\mathbf{lengthOmegaGrid})\) times, so when both arguments are large, the computational demand can be high. By reducing the size of the sequences of the \( \alpha \) and \( \omega \) 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 \( \mathbf{numReps} \) of times, in such a way that, at each repetition, a new two-dimensional grid of \((\alpha, \omega)\) points is created around the previous estimates.

- **Backfitting algorithm**. The argument \( \mathbf{maxiter} \) sets the maximum number of backfitting iterations. Through the argument \( \mathbf{stopFunction} \), 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.

• Restrictions. The arguments betaRestrictions and omegaRestrictions set the equality constraints for the β and ω parameters, respectively. For the unrestricted case, betaRestrictions = 1 : nback and omegaRestrictions = 1 : nback. To add restrictions, ‘integer’ vectors of length m can be passed to these arguments, so that positions with the same numeric value correspond to FMM waves whose parameters, β and/or ω, are forced to be equal. Since restricted fitting can be computationally intensive, especially for constraints on ω parameters, a parallel processing implementation can be used when the argument parallelize = TRUE.

3.2 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            | A m-element ‘numeric’ vector of the estimated FMM wave amplitude parameter(s) A. |
| alpha        | A m-element ‘numeric’ vector of the estimated FMM wave phase translation parameter(s) α. |
| beta         | A m-element ‘numeric’ vector of the estimated FMM wave skewness parameter(s) β. |
| omega        | A 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           | A 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.

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

| Argument                  | Default value | Description                                                                 |
|---------------------------|---------------|-----------------------------------------------------------------------------|
| objFMM                    | no default value | The object of class ‘FMM’ to be plotted.                                    |
| components                | TRUE          | TRUE to display a plot of components.                                       |
| plotAlongPeriods          | TRUE          | 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.                           |

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.

### 3.4 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 $\alpha$.              |
| beta        | no default value | Vector of the FMM wave skewness parameter $\beta$.                        |
| omega       | no default value | Vector of the FMM wave kurtosis parameter $\omega$.                       |
| from        | 0             | Initial time point of the simulated data.                                  |
| to          | $2\pi$        | 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.                    |

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 $2\pi$. 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` and `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`.

### 4 Basic usage of the FMM package

The two examples below, based on FMM synthetic data, illustrate the basic uses and capabilities of the functions implemented in the `FMM` package. The first example is based on a simulated data set from a multicomponent FMM model and illustrates the fitting process and the presentation of its results. The second is devoted to the restricted version of FMM models and serves to exemplify the control of the grid search related arguments to reduce the execution time.
4.1 Example 1: A multicomponent FMM data set

The first simulated data set consists of 100 observations from an FMM$_2$ model with intercept parameter $M = 0$ and two FMM waves defined by the parameters $\nu_1 = (A_1 = 2, \alpha_1 = 1.5, \beta_1 = 0.2, \omega_1 = 0.1)'$ and $\nu_2 = (A_2 = 2, \alpha_2 = 3.4, \beta_2 = 2.3, \omega_2 = 0.2)'$, respectively. 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.

```r
R> library("FMM")
R> set.seed(15)
R> fmm2.data <- generateFMM(M = 0, A = rep(2, 1),
                           alpha = c(1.5, 3.4),
                           beta = c(0.2, 2.3),
                           omega = c(0.1, 0.2),
                           plot = FALSE, outvalues = TRUE, sigmaNoise = 0.3)
```

The estimation of an FMM$_2$ can be performed by setting `nback = 2` in the fitting function `fitFMM()`.

```r
R> fit.fmm2 <- fitFMM(vData = fmm2.data$y, timePoints = fmm2.data$t,
                      nback = 2, showTime = FALSE)
```

When the argument `showProgress = TRUE` (its default value), a progress indicator and information about the stopping criterion of the backfitting algorithm will be displayed on the console. By default, `maxiter` iterations will be forced, but we can use the argument `stopFunction` to modify the criteria. For example, to continue the search until there is an improvement, in terms of explained variability, of less than 1%, the `R2()` function can be used specifying the `difMax = 0.01` argument.

```r
R> fit.fmm2 <- fitFMM(vData = fmm2.data$y, timePoints = fmm2.data$t, nback = 2, showTime = FALSE, maxiter = 5, stopFunction = R2(difMax = 0.01))
```

The results are displayed by the function `summary()`:

```r
R> summary(fit.fmm2)
```

**Title:**
FMM model with 2 components

**Coefficients:**

- **M (Intercept):** -0.2188
  - `A`, `alpha`, `beta`, `omega`
  - FMM wave 1: 1.9615 3.3743 2.2275 0.2053
  - FMM wave 2: 2.2006 1.4930 0.2474 0.0976
Peak and trough times and signals:

| FM wave 1 | t.Upper | Z.Upper | t.Lower | Z.Lower |
|-----------|---------|---------|---------|---------|
|           | 0.4339  | 3.9237  | 5.7250  | 0.0146  |
| FM wave 2 | 5.9656  | 0.1348  | 4.6103  | -4.0110 |

Residuals:

|         | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---------|------|---------|--------|------|---------|------|
|         | -0.80445 | -0.21776 | -0.01272 | 0.00000 | 0.17982 | 0.66971 |

R-squared:

| Wave 1 | Wave 2 | Total |
|--------|--------|-------|
| 0.6907 | 0.2778 | 0.9685 |

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. From the above summary results, we can see that the variance explained by the fitted model is 96.85% and that the FMM waves are labelled in decreasing order according to the $R^2$ value: the explained variance is 69.07% and 27.78% by “FMM wave 1” and “FMM wave 2”, respectively. 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.

```r
R> fitted.fmm2 <- fitted(fit.fmm2)
R> head(fitted.fmm2)
```

| timePoints | fitted |
|------------|--------|
| 1          | 0.00000000 1.251937 |
| 2          | 0.06346652 1.726570 |
| 3          | 0.12693304 2.269928 |
| 4          | 0.19039955 2.826444 |
| 5          | 0.25386607 3.319683 |
| 6          | 0.31733259 3.681024 |

The code given below creates Figure 1:

```r
R> library("RCColorBrewer")
R> library("ggplot2")
R> library("gridExtra")
R> titleText <- "Example 1: two simulated FMM waves"
R> defaultFMM2 <- plotFMM(fit.fmm2, use_ggplot2 = TRUE ,textExtra = titleText) +
theme(plot.margin=unit(c(1,0.25,1.3,1), "cm")) +
ylim(-5, 5)
R> compFMM2 <- plotFMM(fit.fmm2, components = TRUE, use_ggplot2 = TRUE ,
textExtra = titleText) +
theme(plot.margin=unit(c(1,0.25,0.1), "cm")) +
ylim(-5, 5) +
scale_color_manual(values = brewer.pal("Set1", n = 8)[3:4])
R> grid.arrange(defaultFMM2, compFMM2, nrow = 1)
```

The FMM plots can be written in the R `graphics` or `ggplot2` packages. In this example, 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].
4.2 Example 2: A restricted multicomponent FMM data set

A set of 100 observations is generated from an FMM4 model with intercept parameter $M = 3$, amplitude parameters: $A_1 = 4.5$, $A_2 = 3$, $A_3 = 1$ and $A_4 = 1.5$, and phase translation parameters: $\alpha_1 = 1.5$, $\alpha_2 = 4.2$, $\alpha_3 = 2$ and $\alpha_4 = 4.7$. The symmetry parameters are equal in all four FMM waves and, with regard to the kurtosis, pairs of waves are equal. Specifically, the shape parameters satisfy:

$$
\beta_1 = \beta_2 = \beta_3 = \beta_4 = 3 \\
\omega_1 = \omega_2 = 0.01 \\
\omega_3 = \omega_4 = 0.15
$$

Using the function \texttt{generateFMM()}:

```r
R> set.seed(1115) 
R> rfmm.data <- generateFMM(M = 3, A = c(4.5, 3, 1, 1.5), alpha = c(1.5, 4.2, 2, 4.7),
beta = rep(3, 4), omega = c(rep(0.01, 2), rep(0.15, 2)),
plot = FALSE, outvalues = TRUE, 
sigmaNoise = 0.3)
```

The betaRestrictions and omegaRestrictions parameters allow 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 betaRestrictions = c(1, 1, 1, 1) and omegaRestrictions = c(1, 1, 2, 2).

```r
R> fit.rfmm <- fitFMM(vData = rfmm.data$y, timePoints = rfmm.data$t, nback = 4,
betaRestrictions = c(1, 1, 1, 1), 
omegaRestrictions = c(1, 1, 2, 2),
showTime = FALSE)
```

```
|----------------------------------------|----------------------------------------|
|========================================|========================================|
| Stopped by reaching maximum iterations (4 iterations) |
```

```r
R> summary(fit.rfmm)
```

\textbf{Title:}

FMM model with 4 components

\textbf{Coefficients:}
M (Intercept): 1.4781

| Model  | Alpha (β) | Beta (γ) | Omega (ω) |
|--------|-----------|----------|------------|
| FMM wave 1 | 3.9422 | 1.4086 | 2.9996 | 0.0126 |
| FMM wave 2 | 1.5654 | 4.6953 | 2.9996 | 0.1366 |
| FMM wave 3 | 1.0413 | 1.9800 | 2.9996 | 0.1366 |
| FMM wave 4 | 1.8896 | 4.2109 | 2.9996 | 0.0126 |

Peak and trough times and signals:

| Model  | t Upper Z | t Upper t | t Lower Z | t Lower t |
|--------|-----------|-----------|-----------|-----------|
| FMM wave 1 | 4.6240 | 1.3097 | 4.2721 | −6.8088 |
| FMM wave 2 | 1.5732 | −3.7161 | 5.6553 | −6.3382 |
| FMM wave 3 | 5.1411 | −4.7470 | 2.9400 | −6.7395 |
| FMM wave 4 | 1.0711 | −2.5415 | 0.7192 | −6.6951 |

Residuals:

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|------|---------|--------|------|---------|------|
| −0.735225 | −0.217362 | 0.004187 | 0.000000 | 0.186612 | 0.928811 |

R− squared:

| Wave 1 | Wave 2 | Wave 3 | Wave 4 | Total |
|--------|--------|--------|--------|-------|
| 0.3664 | 0.2765 | 0.1656 | 0.1081 | 0.9166 |

Figure 2 displays the two graphical representations of the fitted model: the fitted FMM model and the component FMM plot on the left and right, respectively.

Figure 2: Graphical representation of the fitted restricted FMM model with \( \beta_1 = \beta_2 = \beta_3 = \beta_4 \), \( \omega_1 = \omega_3 \) and \( \omega_2 = \omega_4 \) constraints. A scatter plot of the simulated data along with the fitted model is displayed on the left (default plot). The component plot is shown on the right.

The fitting of more complex signals requires more execution time. By combining the values of the arguments lengthAlphaGrid, lengthOmegaGrid and numReps of fitFMM(), a balance between the computational cost and the accuracy of the estimates can be achieved. In this example, we managed to reduce the execution time by setting half-length sequences of the default ones, for both the lengthAlphaGrid and lengthOmegaGrid arguments, while increasing the number of repetitions from 3 to 5. Note that this configuration even allows us to slightly improve the accuracy of the solution from \( R^2 = 91.66\% \) to \( R^2 = 92.03\% \).

R> fit.rfmm.eff <- fitFMM( yData = rfmm.data$y, timePoints = rfmm.data$t, nback = 4, betaRestrictions = c(1, 1, 1, 1), omegaRestrictions = c(1, 1, 2, 2), showTime = FALSE, lengthAlphaGrid = 24, lengthOmegaGrid = 12, numReps = 5)
Stopped by reaching maximum iterations (4 iterations)

R> summary(fit.rfmm.eff)

Title:
FMM model with 4 components

Coefficients:
M (Intercept): 0.9556
  A  alpha  beta  omega
FMM wave 1: 3.5811 1.4855 3.0122 0.0136
FMM wave 2: 1.4826 4.6988 3.0122 0.1761
FMM wave 3: 1.0190 2.0158 3.0122 0.1761
FMM wave 4: 1.9404 4.2063 3.0122 0.0136

Peak and trough times and signals:
  t. Upper  Z. Upper  t. Lower  Z. Lower
FMM wave 1: 4.6289 0.6142 4.2143 −6.8739
FMM wave 2: 1.5800 −3.9989 5.4040 −5.5034
FMM wave 3: 5.1802 −4.9250 2.7210 −6.7180
FMM wave 4: 1.0664 −2.3001 0.6519 −6.7150

Residuals:
  Min.  1st Qu.  Median   Mean  3rd Qu.   Max.
−0.65126 −0.20547 −0.02922 0.00000 0.20787 0.91315

R-squared:
Wave 1  Wave 2  Wave 3  Wave 4  Total
0.3656  0.2605  0.1822  0.1120  0.9203

5 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.

5.1 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 model is fitted to the Iqgap2 average expressed values as follows:

R> data("mouseGeneExp", package = "FMM")
R> fitGene <- fitFMM(mouseGeneExp, nPeriods = 2, showProgress = FALSE)
R> 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 3.

Figure 3: Igap2 gene expression data along two periods (grey dots); FMM (red line) and COS (blue line) fitted models.

5.2 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].
Standard ECG signals are registered using twelve leads, calculated from different electrode locations, Lead II being the reference one.

The FMM package includes the analysis of a typical ECG beat 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 beat with a sampling frequency of 250Hz. Specifically, the ECG signal from Lead II in the fifth of the thirty annotated beats is analysed. Recordings are publicly available on (http://www.physionet.org). Data are saved as ecgData in the package. For an ECG beat, an FMM_ecg, a fifth order multicomponent FMM model can be fitted with the instruction:

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

Title:
FMM model with 5 components

Coefficients:
M (Intercept): 5.2908

|            | alpha | beta | omega |
|------------|-------|------|-------|
| FMM wave 1 | 0.6387 | 5.5134 | 3.2786 | 0.0322 |
| FMM wave 2 | 0.0995  | 4.4191 | 3.7638 | 0.1365 |
| FMM wave 3 | 0.2412  | 5.3504 | 0.6575 | 0.0320 |
| FMM wave 4 | 0.3310  | 5.5906 | 4.7894 | 0.0122 |
| FMM wave 5 | 0.0668  | 1.7998 | 2.1309 | 0.1641 |

Peak and trough times and signals:

|            | t.Upper | Z.Upper | t.Lower | Z.Lower |
|------------|---------|---------|---------|---------|
| FMM wave 1 | 2.3674  | 6.2328  | 3.2488  | 4.7246  |
| FMM wave 2 | 1.1898  | 4.9486  | 2.0805  | 4.6806  |
| FMM wave 3 | 2.3959  | 6.0793  | 2.1870  | 4.5543  |
| FMM wave 4 | 2.4227  | 5.7845  | 2.4715  | 4.7177  |
| FMM wave 5 | 5.1224  | 4.8646  | 4.3649  | 4.7226  |

Residuals:

|            | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|------------|------|---------|--------|------|---------|------|
|            | −0.0694805 | −0.0096146 | −0.0001619 | 0.0000000 | 0.0098442 | 0.0625571 |

R-squared:

|            | Wave 1 | Wave 2 | Wave 3 | Wave 4 | Wave 5 | Total |
|------------|--------|--------|--------|--------|--------|-------|
|            | 0.7650 | 0.0927 | 0.0559 | 0.0402 | 0.0380 | 0.9919 |

It is worth noting that the FMM package not only provides ECG signal-fitting (the left hand panel in Figure 4), but it also does wave decomposition and fiducial mark annotations on the desired waves (the right hand panel in Figure 4). 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.

5.3 Example 3: Neuroscience

5.3.1 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, 2021a].

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:

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

**Title:**  
FMM model with 2 components

**Coefficients:**

- $M$ (Intercept): 44.9448
  - alpha: 52.9023 4.4160 3.0606 0.0413
  - beta: 18.5044 4.6564 4.9623 0.0322

**Peak and trough times and signals:**

- t.Upper: 1.2777 110.8354 5.9670 -2.5003
- Z.Upper: 1.4320 36.9015 1.5649 -16.2606

**Residuals:**

- Min: -14.3012
- 1st Qu.: -1.0033
- Median: 0.7473
- Mean: 0.0000
- 3rd Qu.: 1.3230
- Max: 24.8616

**R-squared:**

- Wave 1: 0.9064
- Wave 2: 0.0604
- Total: 0.9669

The goodness of fit of the FMM$_2$ model can be ascertained in Figure 5. 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 $R^2 = 0.9669$. 

---

**Figure 4:** 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.
Figure 5: Neuronal AP simulated with the Hodgkin-Huxley model (parameters: $C = 1$, $g_{Na} = 260$, $g_K = 30$, $g_L = 0.31$, $V_K = −12$, $V_{Na} = 115$, $V_L = 10.6$, $\alpha_n = 1.15$, $\beta_n = 0.85$, $\alpha_m = 0.9$, $\beta_m = 1.3$, $\alpha_h = 1$, $\beta_h = 1$ 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.

5.3.2 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, 2021b]. Each AP is modeled by two components. The $\beta$ and $\omega$ parameters are constrained between AP curves. The code below fits the model.

R> data("neuronalAPTrain", package = "FMM")
R> nAPs <- 3; restriction <- c(rep(1,nAPs), rep(2,nAPs))
R> APTrain<-fitFMM(neuronalAPTrain, nback = nAPs*2, 
                    betaRestrictions = restriction, 
                    omegaRestrictions = restriction, 
                    showProgress = FALSE)

R> summary(APTrain)

Title:
FMM model with 6 components

Coefficients:
  M (Intercept): 135.4915
    A   alpha  beta  omega
FMM wave 1:  51.1192  4.2327  2.8190  0.0384
FMM wave 2:  51.7219  6.1358  2.8190  0.0384
FMM wave 3:  52.1105  1.7536  2.8190  0.0384
FMM wave 4: 19.6488 0.0975 4.8644 0.0551
FMM wave 5: 19.2020 1.9978 4.8644 0.0551
FMM wave 6: 20.3642 4.4783 4.8644 0.0551

Peak and trough times and signals:

| Wave       | t.Upper | Z.Upper | t.Lower | Z.Lower |
|------------|---------|---------|---------|---------|
| FMM wave 1 | 1.1036  | 111.2685| 0.6274  | −0.1787 |
| FMM wave 2 | 3.0067  | 111.4205| 2.5304  | −1.2404 |
| FMM wave 3 | 4.9077  | 111.7226| 4.4314  | −1.4963 |
| FMM wave 4 | 3.1109  | 58.2307 | 3.3336  | −13.7196 |
| FMM wave 5 | 5.0112  | 58.2938 | 5.2339  | −13.2943 |
| FMM wave 6 | 1.2086  | 58.5306 | 1.4313  | −14.2673 |

Residuals:

|        | Min. | 1st Qu. | Median | Mean  | 3rd Qu.| Max.   |
|--------|------|---------|--------|-------|--------|--------|
| FMM     | −15.1339 | −1.5071  | 0.4973 | 0.0000 | 1.6303 | 19.9627 |

R-squared:

Wave 1 Wave 2 Wave 3 Wave 4 Wave 5 Wave 6 Total
0.3502 0.2922 0.2485 0.0411 0.0274 0.0244 0.9838

In Figure 6, the fit of the FMM<sub>ST</sub> model can be visualized. The goodness of fit of the model is excellent, achieving an $R^2 = 0.9838$.

Figure 6: Neuronal AP simulated with the Hodgkin-Huxley model (parameters: $C = 1, g_{Na} = 232, g_K = 45, g_L = 0.215, V_K = −12, V_{Na} = 115, V_L = 10.6, a_n = 0.95, b_n = 1.3, a_m = 1, b_m = 1.15, a_h = 1, b_h = 1$ and applying a short square current of 4.5 $\mu$A for 1 millisecond) and the estimated FMM<sub>ST</sub> signal in red. The components plot of the model can be seen on the right hand side of the figure.

6 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.

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 improvement of the computational efficiency. Currently, one of the most important limitations of the stable version of the package (version 0.1.2) is that the fitting process, especially of the restricted model, can be a time-consuming process. Nevertheless, the execution times of the development version available on the GitHub repository (https://github.com/alexARC26/FMM) have already been significantly reduced, in many cases by an order of 10 times or more.

Acknowledgments

This research was partially supported by the Spanish Ministerio de Economía y Competitividad, grant PID2019-106363RB-I00.

References

Anafi RC, Francey LJ, Hogenesch JB, Kim J (2017). “CYCLOPS Reveals Human Transcriptional Rhythms in Health and Disease.” Proceedings of the National Academy of Sciences, 114(20), 5312–5317. doi:10.1073/pnas.1619320114.

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

Bayes de Luna A (2007). Basic Electrocardiography: Normal and Abnormal ECG Patterns. John Wiley & Sons, Ltd. doi:10.1002/9780470692622.

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

Carlucci M, Kriščiūnas A, Li H, Gibas P, Koncevičius K, Petronis A, Oh G (2020). “DiscoRhythm: An Easy-To-Use Web Application And R Package for Discovering Rhythmicity.” Bioinformatics, 36(6), 1952—1954. doi:10.1093/bioinformatics/btz834.

Caro-Martin, Delgado-Garcia, Gruz, Sánchez-Campusano (2018). “Spike Sorting Based on Shape, Phase, and Distribution Features, and K-TOPS Clustering with Validity and Error Indices.” Scientific Reports, 8(1), 1–28. doi:10.1038/s41598-018-35491-4.

Chang W, Cheng J, Allaire J, Xie Y, McPherson J (2020). shiny: Web Application Framework for R. R package version 1.5.0, URL https://CRAN.R-project.org/package=shiny.

Cornelissen G (2014). “Cosinor-Based Rhythmetry.” Theoretical Biology and Medical Modelling, 11, 16. doi:10.1186/1742-4682-11-16.

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

Downs TD, Mardia KV (2002). “Circular Regression.” Biometrika, 89(3), 683–698. doi:10.1093/biomet/89.3.683.

Gerstner W, Kistler WM, Naud R, Paninski L (2014). Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Cambridge University Press. doi:10.1017/CBO9781107447615.

Gierke CL, Helget R, Cornelissen-Guillaume G (2018). CATkit: Chronomics Analysis Toolkit (CAT): Periodicity Analysis. R package version 3.3.3, URL https://CRAN.R-project.org/package=CATkit.

Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE (2000). “PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals.” Circulation, 101(23), E215–20. doi:10.1161/01.cir.101.23.215.

Gouwens NW, Sorensen SA, Berg J, Lee C, et al. (2019). “Classification of Electrophysiological and Morphological Neuron Types in the Mouse Visual Cortex.” Nature Neuroscience, 22, 1182—1195. doi:10.1038/s41593-019-0417-0.
FMM: Frequency Modulated Möbius Models in R

Hodgkin AL, Huxley AF (1952). “A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve.” The Journal of Physiology, 117(4), 500–544. doi:10.1113/jphysiol.1952.sp004764.

Hughes ME, Hegnesch JB, Kornacker K (2010). “JTK_CYCLE: An Efficient Nonparametric Algorithm for Detecting Rhythmic Components in Genome Scale Data Sets.” Journal of Biological Rhythms, 25(5), 372–380. doi:10.3389/jbior.2010.00024.

Kato S, Shimizu K, Shieh GS (2008). “A Circular - Circular Regression Model.” Statistica Sinica, 18(2), 633–645.

Laguna P, Mark RG, Goldberg A, Moody GB (1997). “A Database for Evaluation of Algorithms for Measurement of QT and Other Waveform Intervals in the ECG.” In Computers in Cardiology 1997, pp. 673–676. IEEE.

Larriba Y, Rueda C, Fernández MA, Peddada SD (2018). “A Bootstrap Based Measure Robust to the Choice of Normalization Methods for Detecting Rhythmic Features in High Dimensional Data.” Frontiers in Genetics, 9, 24. doi:10.3389/fgen.2018.00024.

Larriba Y, Rueda C, Fernández MA, Peddada SD (2020). “Order Restricted Inference in Chronobiology.” Statistics in Medicine, 39(3), 265–278. doi:10.1002/sim.8397.

Lynch EP, Houghton CJ (2015). “Parameter Estimation of Neuron Models Using In-Vitro and In-Vivo Electrophysiological Data.” Frontiers in Neuroinformatics, 9, 10. doi:10.3389/fninf.2015.00010.

Mermet J, Yeung J, Naef F (2017). “Systems Chronobiology: Global Analysis of Gene Regulation in a 24-Hour Periodic World.” Cold Spring Harbor Perspectives in Biology, 9(3). doi:10.1101/cshperspect.a028720.

Mosher CP, Wei Y, Kamiński J, Nandi A, Mamelak AN, Anastassiou CA, Rutishauser U (2020). “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. doi:10.1016/j.celrep.2020.02.027.

Moskon M (2020). “CosinorPy: A python Package for Cosinor-Based Rhythmmetry.” BMC Bioinformatics, 21(1), 485. doi:10.1186/s12859-020-03830-w.

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

Nelder JA, Mead R (1965). “A Simplex Method for Function Minimization.” The Computer Journal, 7(4), 308–313. doi:10.1093/comjnl/7.4.308.

Neuwirth E (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2, URL https://CRAN.R-project.org/package=RColorBrewer.

Parsons R, Parsons R, Garner N, Oster H, Rawashdeh O (2019). “CircaCompare: A Method to Estimate and Statistically Support Differences in Mesor, Amplitude and Phase, Between Circadian Rhythms.” Bioinformatics, 36(4), 1208–1212. doi:10.1093/bioinformatics/btz713.

Rácz M, Liber C, Németh E, Fiáth R, Rokai J, Harmati I, Ulbert I, Márton G (2020). “Spike Detection and Sorting with Deep Learning.” Journal of Neural Engineering, 17(1), 016038. doi:10.1088/1741-2552/ab4896.

R Core Team (2020). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Revelle W (2018). psych: Procedures for Psychological, Psychometric, and Personality Research. R package version 1.8.12. URL https://CRAN.R-project.org/package=psych.

Rodríguez-Collado A, Rueda C (2021a). “Electrophysiological and Transcriptomic Features Reveal a Circular Taxonomy of Cortical Neurons.” Preprint on webpage at https://www.biorxiv.org/content/10.1101/2021.03.24.436849v3.

Rodríguez-Collado A, Rueda C (2021b). “A Simple Parametric Representation of the Hodgkin-Huxley Model.” Preprint on webpage at https://www.biorxiv.org/content/10.1101/2021.01.11.426189v1.

Rueda C, Fernandez I, Larriba Y, Rodriguez-Collado A (2021a). “The FMM Approach to Analyze Biomedical Signals: Theory, Software, Applications and Future.” Preprint.

Rueda C, Larriba Y, Lamela A (2021b). “The Hidden Waves in the ECG Uncovered Revealing a Sound Automated Interpretation Method.” Scientific Reports, 11, 3724. doi:10.1038/s41598-021-82520-w.

Rueda C, Larriba Y, Peddada SD (2019). “Frequency Modulated Möbius Model Accurately Predicts Rhythmic Signals in Biological and Physical Sciences.” Scientific Reports, 9(1), 18701. doi:10.1038/s41598-019-54569-1.

Rueda C, Rodríguez-Collado A, Larriba Y (2021c). “A Novel Wave Decomposition for Oscillatory Signals.” IEEE Transactions on Signal Processing, 69, 960—972. doi:10.1109/TSP.2021.3051428.
Sachs M (2014). cosinor: Tools for Estimating and Predicting the Cosinor Model. R package version 1.1, URL https://CRAN.R-project.org/package=cosinor

Shah AS (2020). card: Cardiovascular and Autonomic Research Design. R package version 0.1.0, URL https://CRAN.R-project.org/package=card

Singer JM, Hughey JJ (2019). “LimoRhyde: A Flexible Approach for Differential Analysis of Rhythmic Transcriptome Data.” Journal of Biological Rhythms, 34(1), 5–18. doi:10.1177/0748730418813785

Souza BC, dos Santos VL, Bacelo J, Tort AB (2019). “Spike Sorting with Gaussian Mixture Models.” Scientific Reports, 9(1), 1–14. doi:10.1038/s41598-019-39986-6

Teeter C, Iyer R, Menon V, Gouwens N, Feng D, Berg J, Szafer A, Cain N, Zeng H, Hawrylycz M, Koch C, Mihalas S (2018). “Generalized Leaky Integrate-and-Fire Models Classify Multiple Neuron Types.” Nature Communications, 9(1), 1–15. doi:10.1038/s41467-017-02717-4

Thaben PF, Westermark PO (2014). “Detecting Rhythms in Time Series with RAIN.” Journal of Biological Rhythms, 29(6), 391–400. doi:10.1177/0748730414553029

Wickham H (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org

Zhang R, Lahens NF, Ballance HI, Hughes ME, Hogenesch JB (2014). “A Circadian Gene Expression Atlas in Mammals: Implications for Biology and Medicine.” Proceedings of the National Academy of Sciences, 111(45), 16219–16224. doi:10.1073/pnas.1408886111