Journal Title
XX(X):1–10
©The Author(s) 2025
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

dlmoR: An open-source R package for the dim-light melatonin onset (DLMO) hockey-stick method

Salma M. Thalji[1] and Manuel Spitschan[1,2,3]

Abstract

The dim-light melatonin onset (DLMO) is a commonly used circadian phase marker indicating the start time of evening melatonin synthesis in humans. Several quantitative techniques have been developed to determine DLMO from melatonin time-series, including fixed- or variable-threshold techniques and the hockey-stick method developed by Danilenko and colleagues (2014). Here, we introduce *dlmoR*, an open-source implementation of the hockey-stick method implemented in the R programming language and licensed under the permissive MIT License. Our clean-room implementation followed the algorithm description from the original article, supported by iterative validation against the existing binary executable software. We benchmark our implementation using a total of 112 melatonin time-series data sets from two different studies (Blume et al, 2024; Heinrichs and Spitschan, 2024), and find high agreement between our implementation and the reference implementation. The mean DLMO point discrepancy between implementations was -2.1 ± 11.1 minutes for the Blume et al. (2024) dataset and -1.482 ± 21.7 for the Heinrichs and Spitschan (2024) dataset. Circular correlation coefficients were 0.986 and 0.964, respectively, and paired t-tests ($p > 0.05$) were not significant, suggesting no systematic difference or bias between the methods. *dlmoR* allows for the programmatic and batchable analysis of evening melatonin concentration data, enabling transparency and reproducibility of analytic techniques.

Keywords

Dim Light Melatonin Onset (DLMO), Circadian Rhythm, Biological Clocks, Sleep, Open Source Software, Algorithms

Introduction

The dim-light melatonin onset (DLMO) is the most reliable and widely used marker of circadian phase, serving as a crucial tool in understanding human biological rhythms. By marking the onset of melatonin synthesis under dim light conditions in time, the DLMO provides precise insights into the synchronization of an individual's circadian system with the 24-hour light/dark cycle (Pandi-Perumal et al. 2007; Keijzer et al. 2014). The DLMO is determined through bioanalytic assays of melatonin levels in saliva, plasma, or urine, which are collected under controlled conditions (Benloucif et al. 2008; Pandi-Perumal et al. 2007). The DLMO is instrumental in assessing phase delays, advances, and other deviations from expected circadian timing. Additionally, it plays a key role in research and clinical practice, aiding in the diagnosis and treatment of circadian rhythm sleep disorders, mood disorders and seasonal affective disorder (Pandi-Perumal et al. 2007; Keijzer et al. 2014).

Various methods have been developed to calculate DLMO, including fixed thresholds, dynamic thresholds, and visual estimation. However, these approaches have limitations: fixed thresholds may fail in low melatonin secretors, dynamic thresholds struggle with missing or inconsistent baseline data, and visual estimation is subjective and prone to variability (Kennaway 2023; Glacet et al. 2023). The hockey-stick algorithm (Danilenko et al. 2014) addresses these challenges by modeling melatonin rise as a piecewise linear-parabolic curve. This approach provides a more objective and reliable estimate of DLMO and has demonstrated superior accuracy, repeatability, and robustness compared to traditional methods, making it a recommended option for both research and clinical applications (Danilenko et al. 2014; Glacet et al. 2023).

To make this algorithm accessible to the broader research and clinical community, a software implementation as a Windows program format was developed and released by the authors (Danilenko and Verevkin 2020). However, its usability is limited by the constraints of being a binary executable. This closed-source format does not allow researchers to inspect, modify, or directly interact with the underlying implementation of the algorithm, reducing both flexibility and the ability to fully understand its inner workings. Additionally, executable files without an API do not integrate seamlessly with widely used analytical

[1] Translational Sensory & Circadian Neuroscience, Max Planck Institute for Biological Cybernetics, Tübingen, Germany
[2] Chronobiology & Health, TUM School of Medicine and Health, Technical University of Munich, Munich, Germany
[3] TUM Institute for Advanced Study (TUM-IAS), Technical University of Munich, Garching, Germany

Corresponding author:
Manuel Spitschan, Translational Sensory and Circadian Neuroscience, Max Planck Institute for Biological Cybernetics, Max-Planck-Ring 8-14, 72076 Tübingen, Germany.
Email: manuel.spitschan@tuebingen.mpg.de

workflows, which often require scripting capabilities and compatibility with reproducible pipelines.

This paper introduces the open-source R package *dlmoR* implementing the hockey-stick method, significantly broadening access to advanced DLMO analysis tools. The R package addresses the limitations of the executable format by enabling users to inspect the algorithm's implementation and gain a deeper understanding of its inner workings. It also enhances flexibility, allowing researchers to adapt parameters for specific experimental requirements. By adopting an open-source license and framework, the package fosters collaborative development and community contributions, enabling the addition of new features and improvements over time. Furthermore, its integration into the R programming environment facilitates seamless incorporation into analytical pipelines, guaranteeing usability across varied research workflows. Together, these features make the package accessible, scalable, customizable, and sustainable for a broad range of research applications.

Methods

Algorithm analysis and prior implementation

Description of the algorithm from original article Our open-source R implementation of the hockey-stick algorithm began with a thorough analysis of the original algorithm description provided in Danilenko et al. (2014). The published article offered a high-level understanding of the algorithm's structure and intended functionality. Additionally, we benefited from supplementary insights shared through personal correspondence with the author, which provided a natural language elaboration of some of the algorithm's key components.

Closed-source implementation of the hockey-stick algorithm A Windows implementation of the hockey-stick algorithm was developed by the authors and is available as a free download (Danilenko and Verevkin 2020). This implementation, henceforth referred to as *hockeystickexe*, is only available as a binary executable with no source code, but serves as the reference implementation for this work.

Iterative development

In combination, the original article and personal communication formed the foundation for our analysis and subsequent replication efforts, enabling us to extract algorithmic details and implementation specifications. Using these as our guide, we developed a first implementation of the algorithm in R. To validate and refine our approach, we conducted an iterative comparison of outputs between our R implementation – *dlmoR* – and *hockeystickexe*. Specifically, we input identical datasets into both implementations and systematically analyzed the differences in their outputs. Each discrepancy revealed insights into potential misunderstandings or deviations in our methodology, guiding targeted adjustments and fine-tuning.

This iterative, data-driven process allowed our approach to converge toward the behavior of the original algorithm. Through successive refinements, we ensured that our method accurately reflected our understanding of the algorithm's logic and objectives as presented in the original publication. What we present here is a fully functional implementation grounded in this iterative and comparative methodology.

Implementation details

We chose to implement the hockey-stick algorithm in R (R Core Team 2024) due to its wide use, flexibility, open-source availability cross-platform support and modularity. Our implementation was designed in a modular fashion, with each module corresponding to a distinct component of the framework (see Figure 1). This modular approach facilitates maintainability, testing, and potential future extensions.

Each module was implemented as a separate script or set of scripts, focusing on the following key functionalities:

- *Preprocessing*: Functions handle the input data formatting and any required preprocessing steps, ensuring compatibility with the algorithm.
- *Core algorithm*: The core computation logic of the algorithm is encapsulated in dedicated modules, ensuring clarity and focus on the primary processing tasks.
- *Postprocessing and visualization*: Separate modules generate outputs, including tables and visualizations to aid in interpreting the results.

The implementation utilizes several R packages to streamline development and enhance functionality, including *dplyr* for data manipulation, *tidyverse* for efficient data handling, and *ggplot2* for visualization (Wickham et al. 2023, 2019; Wickham 2016). Custom functions were developed where necessary to meet the specific requirements of the algorithm.

A comprehensive documentation system was embedded within the code, including inline comments and a user guide, to promote reproducibility and ease of use. Additionally, basic unit tests and validation checks were integrated into the workflow to ensure reliability and correctness. Error handling mechanisms, such as logging and exception handling, are also included to provide robust operational stability.

Performance evaluation

To evaluate the performance of the *dlmoR* implementation, its concordance in dim-light melatonin onset (DLMO) estimation was assessed against the hockey-stick executable (*hockeystickexe*).

Datasets and sampling protocols Two datasets comprising a total of 120 melatonin concentration time profiles were analyzed to determine DLMO using both methods.

- *Blume et al. (2024) dataset*: A total of 96 melatonin concentration time profiles, each consisting of 14 data points, were collected from 16 participants, each of whom attended three separate laboratory visits. Saliva samples ($\geq$ 1 mL) were collected using Salivettes (Sarstedt, Nürnberg, Germany) at 30-minute intervals.
- *Heinrichs and Spitschan (2024) dataset*: A total of 24 melatonin concentration time profiles, each consisting of 23 data points, were collected from 12 participants during two-day laboratory visits. Each participant contributed two profiles, with saliva samples ($\geq$ 1 mL) collected at 45-minute intervals.

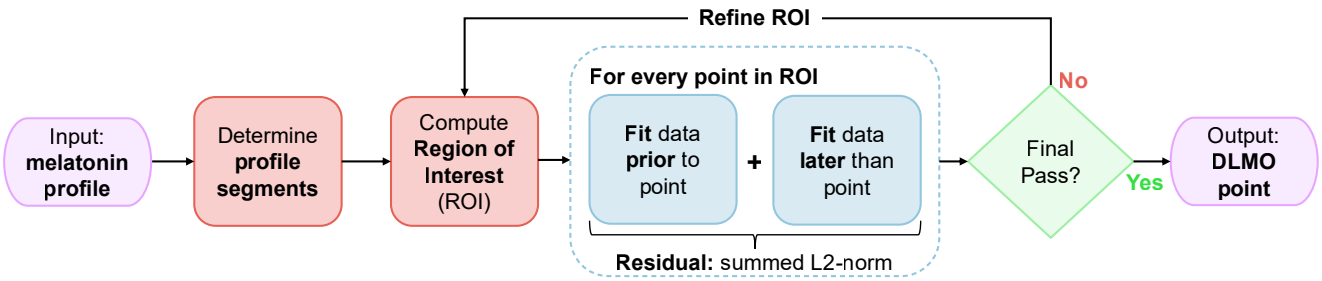

Figure 1. Flowchart of the dlmoR algorithm, which determines the dim light melatonin onset (DLMO) timestamp by identifying the inflection point in a melatonin profile. The algorithm segments raw melatonin concentration data and defines a region of interest (ROI) for detecting melatonin rise. Within this ROI, piecewise fits are applied to data points on either side of each candidate point, and the composite residuals are calculated. After iterating through the ROI, a refined region is established based on the ten percent lowest residuals, and the process is repeated. The point with the minimum residual is identified as the DLMO point.

The exact protocols and research questions asked in these two studies are irrelevant to the purpose of determining the DLMO, and are therefore not further described. In both datasets, melatonin concentrations were quantified using a radioimmunoassay (RIA, RK-DSM2, NovoLytiX GmbH, Switzerland) with a limit of quantification of 0.5–50 pg/mL, a detection limit of 0.2 pg/mL, a mean intra-assay precision of 7.9%, and a mean inter-assay precision of 9.8%.

Thresholds for DLMO analysis For the performance evaluation, we assumed the following ascending level thresholds. For the Blume et al. (2024) dataset, an ascending level threshold of 5 pg/mL, consistent with the methodology outlined in Blume et al. (2024), was applied. For the Heinrichs and Spitschan (2024) dataset, the default ascending level threshold of 2.3 pg/mL was used.

Data exclusion Eight of the 96 profiles in the Blume et al. (2024) dataset did not produce DLMO estimates using either method for one of three reasons: the profile was ambiguous, with no distinct baseline or discernible melatonin rise; the melatonin concentrations of the entire profile remained above the pre-defined threshold; or the entire profile remained below the pre-defined threshold. These data were excluded, leading to the inclusion of 88 time series from the Blume et al. (2024) dataset and 24 time series from the Heinrichs and Spitschan (2024) dataset, yielding a total of 112 datasets included in the analysis.

Analysis workflow The analysis workflow differed between *dlmoR* and *hockeystickexe*. In *dlmoR*, melatonin profiles from both datasets were batch-processed using a custom script. The script parsed all melatonin profiles and determined the DLMO timestamp for each profile based on the user-defined ascending level threshold. Additionally, *dlmoR* saved a data structure for each profile containing the calculated DLMO timestamp, profile fit parameters, and the resulting plot. For analysis with *hockeystickexe*, each of the 112 melatonin profiles was manually entered into the *hockeystickexe* graphical user interface (GUI). The analysis for each profile was initiated by a button click and the resulting plot was saved manually. The DLMO timestamp printed in the GUI was manually tabulated in a spreadsheet alongside the corresponding sample name.

Code, materials and data availability

All code, materials and data is available on our GitHub repository (https://github.com/tscnlab). *dlmoR* is available under the MIT License.

Description of the hockey-stick algorithm

Overall overview

The hockey-stick algorithm comprises three distinct phases (see Fig. 1). First, the input melatonin profile is validated and segmented into baseline and ascending intervals. Next, a region of interest (ROI) is established, within which the dim-light melatonin onset (DLMO) point will be sought in the subsequent phase. Finally, a piecewise function is fitted to the segmented melatonin profile, and the DLMO point is identified as the location within the ROI where the fitted profile yields the lowest residual. This time point is then reported as the output.

In the description that follows, we detail our interpretation of the algorithm, grounded in the original formulation by Danilenko et al. (2014), as well as personal communication with the authors. In this description, *node* refers to a melatonin data point, and *segment* refers to the region between a left node and a right node, including the right node but excluding the left node. More precise implementation details are provided as pseudocode in the supplementary materials.

Validating the melatonin profile

First, the input melatonin profile is validated to ensure that it meets the necessary criteria for DLMO analysis using the hockey-stick method. Profiles are included in the analysis only if they exhibit a rise above a defined threshold (default value: 2.3 pg/mL, which is modifiable). If this criterion is not met, a warning is triggered, indicating the absence of a dynamic portion in the profile ("no dynamic part"). In addition, profiles with only one or two data points are considered insufficient for analysis and a corresponding warning is issued ("insufficient number of data points"). Finally, in cases where the profile shows a rise, but terminates with data points below the threshold, these terminal points are excluded from the analysis.

Determining the profile segments

Defining the base region Any segment below the threshold with a zero or negative slope ($\tan\theta \leq 0$), or crossing the threshold in a downward direction, is classified as part of the *base* section. The *base* section also includes all segments located to the left of this segment. If a segment descends across the threshold or if the difference between the values of the two nearest nodes in the *base* section exceeds half the threshold value, the algorithm flags an inconsistency. In such cases, the following output is generated at the end of the module ("Profile consistency at base part: no").

Defining the ascending region The ascending region is defined as follows:

1. The segment that crosses the ascending threshold, along with all subsequent segments above the threshold, is classified as the *ascending* part. If the profile exhibits two rises above the threshold with an interval of less than two hours (default value), only the segments from the second rise onward are considered as part of the *ascending* section. The interval length is customizable and can be adjusted by the user.
2. In the portion of the profile not labeled as *base*, the *ascending* part also includes the steepest segment, all segments with a steepness exceeding half that of the steepest segment, and any segments situated between these steep segments. This rule does not exclude any segments identified in rule (1) but may add additional segments, including those below the threshold, if they meet the steepness criteria.
3. If the segment immediately preceding the first *ascending* segment (as determined by rules 1 and 2) has a steepness of at least half that of the *ascending* segment, it is also classified as part of the *ascending* section. Rule (3) is then reapplied iteratively to include additional segments if the same condition holds.

Truncating the ascending region Three distinct rules are applied to truncate the *ascending* region, ensuring that the melatonin rise considered for determining the DLMO is both rapid and consistent.

1. First, the rightmost segment of this region is excluded from the *ascending* portion if it has a zero or negative slope ($\tan\theta \leq 0$).
2. The parallelogram rule: After applying rule (1), segments of the *ascending* melatonin profile that rise too slowly are truncated using the properties of an optimal parallelogram fitted around all *ascending* points and the last data point preceding the *ascending* region (see Fig. 2). Specifically, the ratio of the slope of the parallelogram's longest diagonal to the slope of its lateral edge must be greater than 1/2. If this condition is not met, the last segment of the *ascending* profile is truncated. This process is repeated iteratively, with the parallelogram being recalculated after each truncation, until the rule is satisfied.

   The construction of this parallelogram is subject to three constraints: (i) the lower edge must be horizontal and pass through the point with the lowest melatonin level, (ii) the upper edge must be horizontal and pass through the point with the highest melatonin level, and (iii) all *ascending* points must lie within the parallelogram. The term *optimal* refers to a parallelogram with the smallest possible area that satisfies these constraints.

   This construction and optimization process involves determining the lower-left x-coordinate of the parallelogram (p_l), the lower-right x-coordinate (p_r), and the slope of the lateral edge (p_s) (see Fig. 2). By treating these as free parameters, the first and second constraints are inherently satisfied. The third constraint, that all *ascending* points are encompassed by the parallelogram, is treated as a soft constraint. Mathematically, the optimization can be expressed as follows:

$$\arg\max_{p} A(\mathbf{p}) - \lambda C_{(}\mathbf{p}, \mathbf{x}) \tag{1}$$

   where $\mathbf{p} = (p_l, p_r, p_s)$ is the optimization parameters, $\mathbf{x}$ are all datapoints labeled as *ascending* and the last point prior to the first *ascending* point, $A(\mathbf{p})$ and $C(\mathbf{p}, \mathbf{x})$ are the functions that compute the area of the parallelogram and soft-constrained respectively, and $\lambda >> 1$ is a weighting scalar. We solve this optimization problem using the SANN algorithm available in the *Optim* package (Pan 2022) in R. For brevity, the formulation of $A(\mathbf{p})$, $C(\mathbf{p}, \mathbf{x})$, and the initialization procedure are detailed in supplementary pseudocode 4.
3. The rightmost segment cannot be less than half as steep as the steepest of the remaining *ascending* segments (as defined after applying truncation Rules 1 and 2).

Rules (2) and (3) can be disabled if desired.

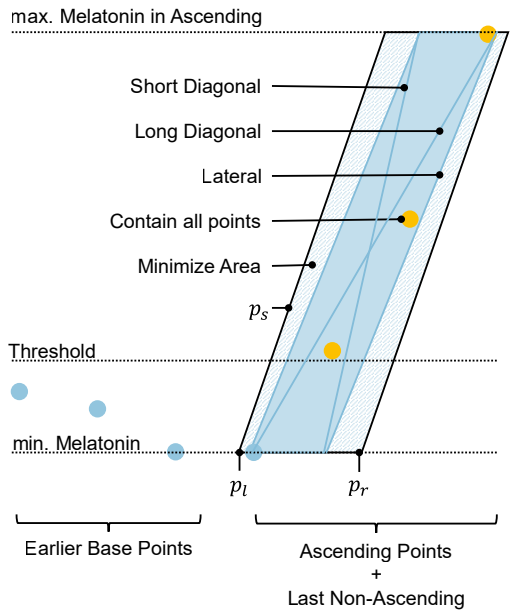

Figure 2. An illustration of the properties of a parallelogram used in the *parallelogram rule* for *ascending* region truncation. Annotations indicate the properties of an optimal (i.e. smallest area) parallelogram fitted around all *ascending* points and the last data point preceding the *ascending* segment.

Adding a segment to the left of the ascending region If at this stage all segments belong to the *ascending* region , and the leftmost node is the only one that lies below half the threshold, an additional node is added 30 minutes to the left of it with a melatonin concentration level equal to either two times lower than the level of the leftmost node or zero, whichever is higher. The added segment is classified as *intermediate* (see below).

Truncating the base region If the *base* part begins with nodes lying above the threshold, these nodes are excluded from the *base* part (see Fig. 4, Panel A). Although they remain visible in the figure, they are not included in the analysis.

Defining the intermediate region Any segments lying between the *base* and *ascending* regions, that belong to neither of the two, are classified as *intermediate*.

Computing the bounds of the ROI

The region of interest (ROI) refers to the portion of the melatonin profile that will be parsed in the subsequent stage of the algorithm to identify the DLMO point. First, a line of interest is defined for the left and right bounds of the ROI, and then the upper and lower bounds are defined.

Line of interest To determine the line of interest, there are two scenarios:

1. *No intermediate point present*: If the *base* and *ascending* parts are defined, and no *intermediate* segments are present, the line of interest includes the last half (50%) of the final *base* segment and the first 95% of the first *ascending* segment. This interval is projected onto the time-axis and is denoted by a thick line parallel to the x-axis on the output plot (see Fig. 3). If there is only a single *base* node, the line of interest becomes narrower, spanning from this node to the first 95% of the first *ascending* segment.
2. *Intermediate points present*: If *intermediate* segments are present, the line of interest spans from immediately after the last *base* node, includes all *intermediate* segments, and extends to the first 95% of the first *ascending* segment.

ROI definition The ROI is defined by the bounds of the line of interest, which set its left and right limits, while the upper and lower boundaries are determined by the threshold value and the melatonin level of the lowest node in the profile, respectively (see Fig. 3).

Find the DLMO point

At this stage, all data points have been categorized into two (or three) groups: *base*, *ascending*, and (when present) *intermediate*. Additionally, an ROI has been defined within which we aim to identify the inflection point. Now, our method employs two phases to detect the inflection point: a coarse phase and a refinement phase (see Fig. 3).

Coarse phase of DLMO search In the coarse phase, we perform a grid search over the ROI. The search uses step sizes of 0.1 in the time dimension (x-values) and 0.2 in the y-values. For each point of interest (POI) in the grid, the following steps are carried out:

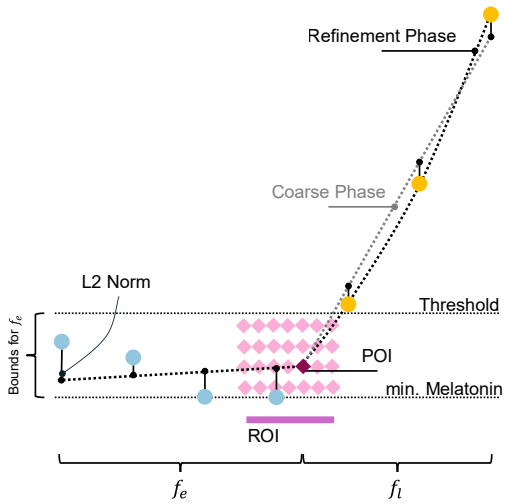

Figure 3. Schematic of the iterative grid search for the DLMO point. The *coarse phase* iterates over the region of interest (ROI), fitting piecewise functions to the *base* (f_e) and *ascending* (f_t) profile regions and calculating a composite residual for each point of interest (POI). The *refinement phase* narrows the ROI with finer resolution, and the POI with the lowest residual is identified as the DLMO time point.

1. *Linear fit to earlier points*: A linear model is fit that minimizes the L_2-norm for all points occurring earlier than the POI, including the POI itself. The linear fit is constrained to pass through the POI, and its slope is limited to the range of -0.2 to 0.2. Additionally, the fitted line cannot exceed the highest y-value (threshold) or fall below the lowest y-value in the fitting domain.
2. *Linear fit to later points*: Similarly, a linear model is fit that minimizes the L_2-norm for all points occurring later than the POI, including the POI itself. This line is also constrained to pass through the POI.

The overall cost associated with each POI is calculated as the sum of the residual L_2-norms from the two linear fits.*

Refinement phase of DLMO search In the refinement phase, we narrow the ROI to focus on the extrema of the 10% of points with the lowest costs from the coarse phase. This phase is similar in approach but uses finer step sizes of 0.01 in both dimensions to achieve greater precision (see Fig. 4).

During the refinement phase, a parabola is fit to the later points. This parabola must satisfy the following constraints: (i) the parabola is required to pass through the POI, (ii) the derivative of the parabola must never be zero, (iii) the slope of the parabola at the POI must be at least half as steep as the slope of the linear fit identified during the coarse phase and (iv) steeper than the slope of the linear fit for the earlier points.

*The description of this portion of the algorithm in the publication by Danilenko et al. (2014) differed from the natural-language explanation provided in personal correspondence, in regards to the use of weighting for the L_2-norms of the fits based on *base* labels. In our *dlmoR* implementation, we chose not to apply any weighting, and after conducting multiple trials, we found no evidence that weighting improved the results.

Generally, for both the coarse and refinement phases, the cost of a point can be expressed as:

$$R = f_e(p_t, \{x_t : x_t \leq p_t\}) + f_l(p_t, \{x_t : x_t \geq p_t\}) \quad (2)$$

where x_t represents all data points and p_t is the point of interest (POI). Specifically:

- $\{x_t : x_t \leq p_t\}$ denotes all points earlier than and/or equal to the POI.
- $\{x_t : x_t \geq p_t\}$ denotes all points later than and/or equal to the POI.

Here, f_e is the linear fit function applied to the earlier data points, and f_l is the fit function for the later data points, which can be either linear or parabolic depending on the search phase. Details of this implementation can be found in supplementary pseudocode 9.

By running a coarse and refined search phase, the method ensures a systematic and precise identification of the DLMO point within the defined ROI.

Results

Clean-room implementation of the hockey-stick algorithm

Despite the original hockey-stick software being only available as a Windows executable, we were able to derive an R-based clean-room implementation of the hockey-stick algorithm developed by (Danilenko et al. 2014).

Performance evaluation

We assessed the concordance in dim light melatonin onset (DLMO) time estimates derived from the two methods – *dlmoR* and *hockeystickexe* – applied to two datasets comprising a total of 120 melatonin time-series Blume et al. (2024); Heinrichs and Spitschan (2024), with a total of 112 usable datasets. Circular statistics were employed to evaluate concordance due to the periodic nature of time data.

Descriptive Statistics The circular difference in DLMO time estimates between the two methods were summarized using descriptive statistics, namely the mean and standard deviation, to describe the central tendency and dispersion. The mean difference in DLMO estimates between *dlmoR* and *hockeystickexe* is 1.165±28.5 minutes for the Blume et al. (2024) dataset, and -1.482±21.7 minutes for the Heinrichs and Spitschan (2024) dataset.

Outliers Outliers were identified by manually inspecting data profiles with DLMO estimate discrepancies between methods exceeding ±1 hour. Of the 112 profiles analyzed, two significant outliers were observed: a DLMO estimate difference of approximately +80 minutes and one of approximately +240 minutes, indicating that the *dlmoR* method estimated DLMO timestamps more than one hour later than *hockeystickexe*. Close inspection of each scenario highlighted two key scenarios in which *dlmoR* and *hockeystickexe* analysis differs: (1) the handling of spurious above-threshold melatonin rises occuring early in the time profiles, and (2) differing interpretations of multiple rises above threshold between methods. In the case of the melatonin profile with a between-method DLMO difference of 80 minutes, a single melatonin sample briefly rose above the threshold, followed by a decline and a subsequent steady rise within one hour. The *hockeystickexe* algorithm fit the earlier single-point rise, whereas *dlmoR* fit the later steady consecutive rise. For the melatonin profile with a +240 minute difference, a spurious single-point rise near the beginning of the profile (second point in a 14-point time series) was fit as the DLMO by *hockeystickexe*. *dlmoR* disregarded this as noise and fit the steady rise occurring four hours later. Without outliers, the mean discrepancy in DLMO estimate between methods was -2.082 ±11.1 minutes for the Blume et al. (2024) dataset. The small mean differences in both datasets suggest minimal overall difference in DLMO estimate between the methods. The larger standard deviation in the Blume et al. (2024) dataset reflects greater variability in the differences compared to the Heinrichs and Spitschan (2024) dataset.

Method agreement within specific time thresholds The agreement between methods was further assessed by categorizing DLMO estimates based on time difference thresholds. For the Blume dataset, 46 estimates (52.3 %) were within ±5 minutes of each other, 69 estimates (78.4%) were within ± 15 minutes, and the difference in three estimates (3.41%) exceeded ±30 minutes. For the Heinrichs dataset, 16 estimates (66.7%) were within ±5 minutes, 18 estimates (75%) were within ±15 minutes, and the difference in 4 estimates (16.7%) exceeded ±30 minutes (see Fig. 5, Panel B) .

Concordance and bias detection Circular correlation coefficients and paired t-tests were used to evaluate the strength and significance of agreement and to identify systematic biases between the two methods. Circular correlation values were 0.986 (0.911 with outliers) for the Blume et al. (2024) dataset and 0.964 for the Heinrichs and Spitschan (2024) dataset, indicating strong agreement between the methods in both datasets (see Fig. 5, Panel A). Circular paired t-tests yielded p-values of 0.088 (0.601 with outliers) for the Blume dataset and 0.742 for the Heinrichs dataset, suggesting no systematic difference or bias between the methods.

Skewness and bias Analysis of skewness revealed minor biases in DLMO estimation. The Blume et al. (2024) dataset had a skewness of -0.22 (6.37 with outliers), while the Heinrichs and Spitschan (2024) dataset had a skewness of -0.796. These values suggest that *dlmoR* tends to provide slightly earlier estimates compared to *hockeystickexe*.

Visual inspection and residual analysis Visual inspection of fitted melatonin profiles revealed two primary contributors of discrepancies between methods, aside from identified outliers:

1. *Parallelogram-based ascending segment truncation*: The methods differed in the truncation of *ascending* segments, leading to the fitting of different sets of points. This impacted the distribution of residual heatmaps and the selection of the DLMO point, which corresponds to the timepoint with the lowest fit residual.

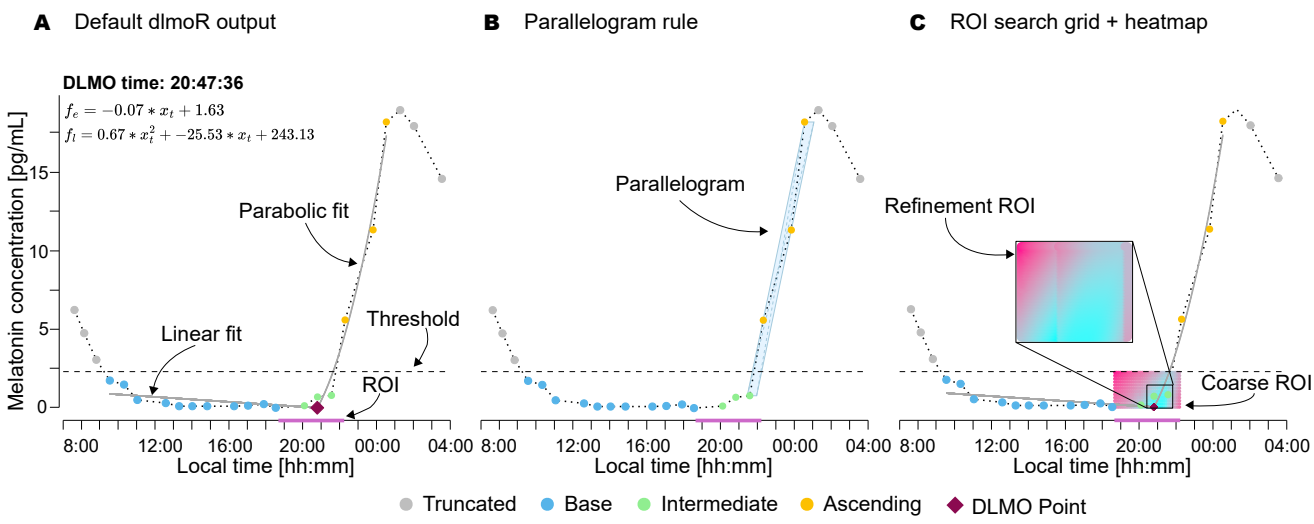

Figure 4. Visualizations of the hockey-stick method implemented in *dlmoR* with an example profile from the Heinrichs and Spitschan (2024) dataset. **A** Default *dlmoR* output, indicating various components of the fitted profile, including annotations of the linear (f_e) and parabolic (f_l) fit parameters. **B** Output including the visualization of the optimal parallelogram used for truncation of the ascending region of the profile. **C** Output with an overlay of the ROI fit residual heatmap used for DLMO point search.

2. *Diffuse region of interest (ROI) heatmap*: The *dlmoR* visualization indicated diffuse ROI heatmaps with multiple points offering similarly good fits (differences on the order of 1 $\times 10^{-2}$). Minor variations in fit procedures likely contributed to residual variation. When small deviations in the fits lead to minor differences in the residuals, a diffuse residual heatmap can correspond to differences in DLMO estimates of 15 minutes or more (see example in Fig. 4, Panel C).

R package

dlmoR is an R package developed to provide a streamlined and efficient tool for analyzing dim light melatonin onset. The package is designed to handle both individual and batch processing of data profiles and offers a range of customizable features to meet diverse user needs.

Key functionalities of the dlmoR *package* The core functionalities of the *dlmoR* package are:

- *Data visualization*: Generates detailed plots that include the following features (see Fig. 4):
  - The raw data profile
  - The subset of points analyzed
  - The identified DLMO timestamp (labeled on the plot and output as an annotation)
  - The two segments of the hockey-stick fit with annotations of their corresponding parameters
  - A time segment highlighting the region of interest for inflection point detection
  - A residual heatmap for points within the region of interest grid, providing insight into the confidence or definitiveness of the result
  - A parallelogram indicating the decision boundary used to trim the points of melatonin rise, offering a clear understanding of how the subset of rise points for the fit was determined
- *Customizability*: Allows users to:
  - Set the ascending threshold
  - Set the multi-rise interval length
  - Modify plot features for tailored visualization
- *Batch processing*: Supports the analysis of multiple data profiles in a single run, enhancing efficiency for large datasets.
- *Automated Outputs*: Automatically saves all key results, including:
  - The raw data profile
  - Labeled segments
  - Fit parameters
  - DLMO point
  - Residual heatmap
  - The generated visualization plot

The residual heatmap is a particularly valuable feature for assessing output quality. Definitive results correspond to heatmaps with a single distinct node and a relatively low residual, while diffuse heatmaps indicate multiple timestamps with similar residuals, suggesting a less definitive outcome.

With its robust capabilities, clear visualizations, and well-documented structure, the *dlmoR* package offers a transparent and intuitive approach to analyzing melatonin data. Its design allows users to closely examine the underlying processes and decision points within the analysis, ensuring clarity and reproducibility in the results.

Discussion

The high concordance between *dlmoR* and *hockeystickexe*, supported by high circular correlations and non-significant paired t-tests, demonstrates robust agreement between methods. Descriptive statistics revealed small mean differences between the methods, with standard deviations indicating greater variability in the Blume et al. (2024) dataset compared to the Heinrichs and Spitschan (2024) dataset. Additionally, time threshold assessments showed that a majority

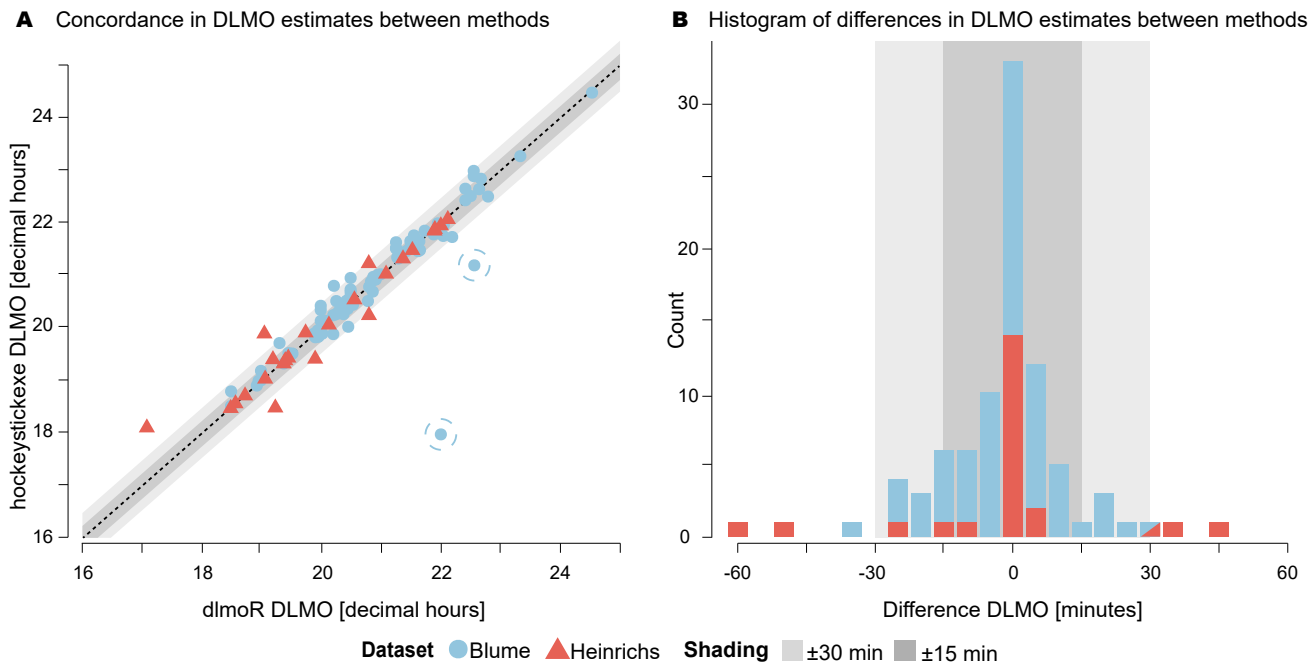

Figure 5. Performance evaluation of *dlmoR* against *hockeystickexe*. **A** Concordance in DLMO estimate between implementations, with two outliers highlighted by circles. **B** Histogram of differences in DLMO estimates between methods

of DLMO estimates fell within 15 minutes of each other, further highlighting the reliability of both methods. Discrepancies can be attributed to differences in the handling of ascending segment truncation and ROI heatmap diffusion. Identified outliers underscore the importance of careful interpretation of the results and highlight the necessity for users to pre-assess raw data profiles for spurious melatonin profiles before running the analysis. This reinforces the principle that the quality of the input data critically influences the reliability of the results. By providing users with access to the underlying ROI heatmap, a feature specific to *dlmoR*, we aim to offer insight into the robustness of the ROI estimate.

The development of the hockey-stick algorithm by Danilenko et al. (2014) was a key step forward in providing an objective framework for determining DLMO, addressing the subjectivity inherent in earlier approaches. However, upon closer examination, this method is not as definitively objective as it might initially appear. The sensitivity of its optimization process—particularly to initialization and local minima—reveals underlying challenges, especially in complex tasks like the multi-constrained optimization of the parallelogram. This sensitivity, coupled with the diffuse nature of the ROI heatmap, highlights that the solutions derived are not singular or definitive but instead represent a range of plausible possibilities.

Rather than masking this inherent uncertainty, we propose a shift toward approaches that maintain objectivity while transparently representing the spread of potential solutions. The diffuse heatmaps often found in our test suggest that the DLMO is not a precise timestamp but rather a region of highest likelihood. A probabilistic approach, such as a Gaussian process model (GPM) (Rasmussen and Williams 2006), could better capture this uncertainty, modeling DLMO as a probabilistic distribution over a plausible range rather than a single optimal point. GPMs offer several advantages in this context. By representing melatonin profiles as smooth, non-linear functions, they provide flexibility and adapt directly to the data without imposing rigid assumptions like breakpoints. They also naturally account for uncertainty by providing credible intervals for potential DLMO regions, offering a more transparent and nuanced view of melatonin onset. Additionally, GPMs avoid the pitfalls of optimization-based methods, such as sensitivity to initialization or local minima, by calculating distributions over plausible solutions, making them particularly well-suited for noisy or complex datasets.

Alternatively, generalized additive mixed models (GAMMs) (Pedersen et al. 2019) provide another flexible, data-driven approach for estimating DLMO. Unlike Gaussian process models, which take a fully probabilistic approach, GAMMs model melatonin changes as smooth, continuous curves that adapt to the underlying patterns in the data while incorporating fixed and random effects. GAMMs are particularly well-suited to capturing gradual or complex trends without relying on abrupt transitions, such as breakpoints in piecewise models. They also account for individual variability through random effects, allowing the model to adjust for differences across individuals, such as chronotype variation, while still identifying shared patterns. Additionally, GAMMs can pinpoint the sharpest increase in melatonin levels using derivatives of the smooth function, providing a precise and objective estimate of DLMO without relying on predefined thresholds.

As we continue to develop dlmoR, we aim to extend it with more flexible and robust tools for DLMO estimation, including both Gaussian process models and GAMMs. These approaches promise to enhance dlmoR's ability to handle noisy or complex datasets while providing more transparent and nuanced representations of melatonin onset.

A critical component of these developments is ensuring the reproducibility and accessibility of our tools. Reproducibility is a cornerstone of scientific research, ensuring that findings can be validated and extended by others (Harrison et al. 2021). However, many scientific software tools lack the transparency, accessibility, and adaptability necessary to meet these standards, often hindering broader use and integration into diverse research workflows (Wilson et al. 2017; Harrison et al. 2021). To address these challenges, our open-source R package promotes reproducible large-scale analyses and facilitates community-driven contributions. Unlike the existing standalone .exe software, which limits modifiability and flexibility, this package aligns with the principles of findability, accessibility, interoperability, and reusability (FAIR), fostering an ecosystem for collaborative development and innovation (Lee et al. 2021; Wilson et al. 2017). Moreover, by adhering to modular design, emphasizing readability, and implementing practices such as version control, continuous integration, and detailed documentation, our package ensures both immediate usability and long-term sustainability (Lee et al. 2021; Harrison et al. 2021). By releasing this package, we aim to empower researchers with a robust tool for melatonin analysis while providing a foundation for future advancements in analytical methodologies.

Conclusion

In this article, we introduced the *dlmoR* package, an open-source R implementation of the hockey-stick algorithm (Danilenko et al. 2014) licensed under the MIT license. Our clean-room implementation produces results comparable to the original, closed-source hockey-stick Windows binary executable. Additionally, we identified several uncertainties in the fitting of the hockey-stick profile, prompting further investigation. The *dlmoR* package will be instrumental in the reproducible analysis of evening melatonin time-series, and can be expanded to include future quantitative algorithms for the estimation of DLMO timing.

Acknowledgements

We wish to thank Dr. Konstantin (Kostya) Danilenko (1962-2023) (Putilov 2023) for detailed discussions of the algorithm and providing a natural-language description of its implementation.

References

Benloucif S, Burgess HJ, Klerman EB, Lewy AJ, Middleton B, Murphy PJ, Parry BL and Revell VL (2008) Measuring melatonin in humans. *Journal of Clinical Sleep Medicine* 4(1): 66–69.

Blume C, Cajochen C, Schöllhorn I, Slawik HC and Spitschan M (2024) Effects of calibrated blue–yellow changes in light on the human circadian clock. *Nature Human Behaviour* 8(3): 590–605.

Danilenko K and Verevkin EG (2020) Hockey-stick software. URL https://neuronm.ru/R&D/hockey-stick.

Danilenko KV, Verevkin EG, Antyufeev VS, Wirz-Justice A and Cajochen C (2014) The hockey-stick method to estimate evening dim light melatonin onset (DLMO) in humans. *Chronobiology International* 31(3): 349–355.

Glacet R, Reynaud E, Robin-Choteau L, Reix N, Hugueny L, Ruppert E, Geoffroy PA, Kilic-Huck Ü, Comtet H and Bourgin P (2023) A comparison of four methods to estimate dim light melatonin onset: a repeatability and agreement study. *Chronobiology International* 40(2): 123–131.

Harrison S, Dasgupta A, Waldman S, Henderson A and Lovell C (2021) How reproducible should research software be? URL https://zenodo.org/records/4761867.

Heinrichs HS and Spitschan M (2024) Within-subjects ultra-short sleep-wake protocol for characterising circadian variations in retinal function. *medRxiv* : 2024–02.

Keijzer H, Smits MG, Duffy JF and Curfs LM (2014) Why the dim light melatonin onset (DLMO) should be measured before treatment of patients with circadian rhythm sleep disorders. *Sleep Medicine Reviews* 18(4): 333–339.

Kennaway DJ (2023) The dim light melatonin onset across ages, methodologies, and sex and its relationship with morningness/eveningness. *Sleep* 46(5): zsad033.

Lee G, Bacon S, Bush I, Fortunato L, Gavaghan D, Lestang T, Morton C, Robinson M, Rocca-Serra P, Sansone SA et al. (2021) Barely sufficient practices in scientific computing. *Patterns* 2(2).

Pan Y (2022) *roptim: General Purpose Optimization in R using C++*. DOI:10.32614/CRAN.package.roptim. R package version 0.1.6.

Pandi-Perumal SR, Smits M, Spence W, Srinivasan V, Cardinali DP, Lowe AD and Kayumov L (2007) Dim light melatonin onset (DLMO): a tool for the analysis of circadian phase in human sleep and chronobiological disorders. *Progress in Neuro-Psychopharmacology and Biological Psychiatry* 31(1): 1–11.

Pedersen EJ, Miller DL, Simpson GL and Ross NE (2019) Hierarchical generalized additive models in ecology: An introduction with mgcv. *PeerJ* 7: e6876. DOI:10.7717/peerj.6876.

Putilov AA (2023) Obituary for Dr. Konstantin Danilenko (19.03.1962–18.01.2023). *Clocks & Sleep* 5(1): 94–97. DOI:10.3390/clockssleep5010010.

R Core Team (2024) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rasmussen CE and Williams CK (2006) *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press. URL http://www.gaussianprocess.org/gpml/.

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

Wickham H, Averick M, Bryan J, Chang W, McGowan L, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel D, Spinu V, Takahashi K, Dunnington D, Wilke C, Woo K and Yutani H (2019) Welcome to the tidyverse. *Journal of Open Source Software* 4(43): 1686. DOI:10.21105/joss.01686. URL https://doi.org/10.21105/joss.01686.

Wickham H, François R, Henry L and Müller K (2023) *dplyr: A Grammar of Data Manipulation*. URL https://dplyr.tidyverse.org. R package version 1.1.2.

Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L and Teal TK (2017) Good enough practices in scientific computing. *PLoS Computational Biology* 13(6): e1005510.

Pseudocode of the hockey-stick algorithm implemented in dlmoR

Algorithm 1: Define *base* region of the melatonin profile

Input: Time series data containing melatonin concentrations $M[i]$, timestamps $t[i]$ (in POSIXct format), and a threshold value τ (default = 2.3 pg/mL)

Output: Annotated time series with base segment labeled $B[i]$ (1 for base, 0 for non-base)

Step 1: Preprocess Data;

Sort the time series data chronologically;

Convert timestamps from POSIXct format to decimal hours to create a continuous numerical representation of time;

Step 2: Compute Profile Slopes;

Calculate the slope between consecutive data points. The slope at point i is defined as:

$$\text{slope}[i] = \frac{M[i] - M[i-1]}{t[i] - t[i-1]},$$

where $M[i]$ is the melatonin concentration and $t[i]$ is the time in decimal hours.

Step 3: Identify Base Segments;

For each point in the time series:

- Label as **base** (`1`) if:

$$\text{slope}[i] \leq 0 \quad \text{and} \quad \big(M[i] \leq \tau \text{ or } M[i+1] \leq \tau\big),$$

  where $M[i+1]$ is the concentration at the subsequent point.
- Otherwise, label as **non-base** (`0`).

Step 4: Propagate Base Labels;

Propagate the base segment label (`1`) backward from the rightmost identified base point. A point i is labeled as base if:

$$B[i] = \begin{cases} 1 & \text{if } i \leq \max\{j \mid B[j] = 1\}, \\ 0 & \text{otherwise.} \end{cases}$$

This ensures that all points preceding the last base point are also labeled as base.

Algorithm 2: Define *ascending* region of the melatonin profile

Input: Time series data containing melatonin concentrations $M[i]$, timestamps $t[i]$, slopes slope$[i]$, and *base* labels; threshold value τ (default = 2.3 pg/mL); interval limit Δt_{limit} (default = 2 hours)

Output: Annotated time series with ascending segments labeled $A[i]$ (1 for ascending, 0 for non-ascending)

Step 1: Detect Transitions Above Threshold;

Identify transitions where melatonin concentrations rise above the threshold τ after being below or equal to it. The transition indicator is defined as:

$$T[i] = \begin{cases} 1 & \text{if } M[i] > \tau \text{ and } M[i-1] \leq \tau, \\ 0 & \text{otherwise.} \end{cases}$$

These transitions mark the starting points of potential ascending segments.

Step 2: Group Consecutive Points into Rise Groups;

Assign a unique group index $G[i]$ to each set of consecutive points that belong to the same rise above the threshold. This is determined using:

$$G[i] = \begin{cases} G[i-1]+1 & \text{if } T[i] = 1, \\ G[i-1] & \text{if } G[i-1] \neq \text{NA}, \\ \text{NA} & \text{otherwise.} \end{cases}$$

Here, a new group is started when a transition is detected ($T[i] = 1$), and points remain in the same group if they are consecutive.

Step 3: Validate Rise Groups Based on Interval Limit;

Filter the rise groups based on the time interval between their starting points. A rise group g is considered valid if:

$$V[g] = \begin{cases} 1 & \text{if } g = 1 \text{ or } \Delta t_g \geq \Delta t_{\text{limit}}, \\ 0 & \text{otherwise.} \end{cases}$$

where:

$$\Delta t_g = t_g - t_{g-1}.$$

This ensures that only rise groups separated by a sufficient time interval are retained.

Step 4: Assign Ascending Labels;

Label points within valid rise groups as ascending if their melatonin concentration exceeds the threshold:

$$A[i] = \begin{cases} 1 & \text{if } G[i] \in \mathcal{V} \text{ and } M[i] > \tau, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $\mathcal{V}$ is the set of valid rise groups where $V[g] = 1$.

Step 5: Identify Steepest Segments;

Identify the steepest slope in the profile and label points as ascending if their slope is at least half the maximum slope in the profile:

$$S[i] = \begin{cases} 1 & \text{if slope}[i] \geq \frac{1}{2} \cdot \text{slope}_{\max}, \\ 0 & \text{otherwise.} \end{cases}$$

This step ensures that significant changes in the profile are captured even if they don't correspond directly to a rise above the threshold.

Step 6: Backward Propagation of Ascending Labels;

Propagate the ascending label backward from the first ascending point to previous points if their slopes are sufficiently steep. For any preceding point j of the first ascending point i:

$$A[j] = \begin{cases} 1 & \text{if } S[j] \geq \frac{1}{2} S[i] \text{ and } A[j+1] = 1, \\ 0 & \text{otherwise.} \end{cases}$$

This process is iterative and stops when the slope condition is no longer satisfied or the start of the series is reached.

return *Annotated time series with $A[i]$ (1 for ascending, 0 for non-ascending)*;

Algorithm 3: Truncate *ascending* region

Input: Time series data containing melatonin concentrations $M[i]$, timestamps $t[i]$, slopes slope$[i]$, and ascending labels $A[i]$.

Output: Updated time series with truncated ascending segment and optimal parallelogram fit parameters.

Step 1: Check for Ascending Segment;

Extract the ascending segment:

$$A_{\text{segment}} = \{i \mid A[i] = 1\}.$$

if $A_{segment}$ *is empty* **then**

  Issue a warning: "No ascending segments found in the profile data.";

  return *Input data unchanged.*

end

Step 2: Find Steepest Slope in Ascending Segment;

Compute the maximum slope in the ascending segment:

$$\text{max_slope} = \max(\text{slope}[i] \mid i \in A_{\text{segment}}).$$

Step 3: Iteratively Truncate Ascending Segment;

Truncate the ascending segment until all rules are satisfied:

- **Rule 1: Positive rightmost slope.** The slope between the last two points in the ascending segment must be positive:

$$\text{rightmost_slope} = \frac{M[i_{\text{last_asc}}] - M[i_{\text{last_asc}-1}]}{t[i_{\text{last_asc}}] - t[i_{\text{last_asc}-1}]} > 0.$$

- **Rule 2: Valid parallelogram fit.** Ensure the parallelogram fit (Algorithm 4) does not violate constraints, such as diagonal slope rules.

while *either rule is violated* **do**

  Identify the last ascending index:

$$i_{\text{last_asc}} = \max\{i \mid A[i] = 1\}.$$

  Set the last point in the ascending segment to non-ascending:

$$A[i_{\text{last_asc}}] \leftarrow 0.$$

  Recompute max_slope.

end

Step 4: Finalize Parallelogram Fit;

Fit a parallelogram to the updated profile data using Algorithm 4:

$$\text{parallelogram_fit(profile_data)}.$$

return *Updated profile data and parallelogram fit parameters.*

Algorithm 4: Fit optimal parallelogram to enclose points with minimal area

Input: Coordinates of points $x[i]$, $y[i]$ in the Cartesian plane.
Output: Optimal parallelogram parameters: x_0 (left edge), x_1 (right edge), and slope.

Step 1: Define Initial Variables;

Compute vertical bounds of the parallelogram:

$$y_0 = \min(y[i]), \quad y_1 = \max(y[i]).$$

Set initial guesses for optimization:

- $x_0^{\text{initial}} = x[1]$ (first point's x coordinate),
- $x_1^{\text{initial}} = x[N]$ (last point's x coordinate),
- Slope:

$$\text{slope}^{\text{initial}} = \frac{y_1 - y_0}{x_1^{\text{initial}} - x_0^{\text{initial}}}.$$

- Adjust x_1 based on slope and vertical difference:

$$x_1^{\text{adjusted}} = x_1^{\text{initial}} - \frac{y_1 - y_0}{\text{slope}^{\text{initial}}}.$$

- Initial guess for optimization:

$$\text{initial_guess} = \left(x_0^{\text{initial}} \cdot 0.8,\ x_1^{\text{adjusted}} \cdot 1.1,\ \text{slope}^{\text{initial}}\right).$$

Step 2: Define Parallelogram Corners;

Compute the corners of the parallelogram based on x_0, x_1, y_0, y_1, and slope:

$$\begin{aligned} \text{Lower Left (LL):} &\quad (x_0, y_0), \\ \text{Lower Right (LR):} &\quad (x_1, y_0), \\ \text{Corners: Upper Left (UL):} &\quad \left(x_0 + \frac{y_1 - y_0}{\text{slope}}, y_1\right), \\ \text{Upper Right (UR):} &\quad \left(x_1 + \frac{y_1 - y_0}{\text{slope}}, y_1\right). \end{aligned}$$

Step 3: Define Constraints;

Enforce constraints to ensure all points $(x[i], y[i])$ lie within the parallelogram:

$$\begin{aligned} c_{\text{upper}}[i] &= y_1 + \text{slope} \cdot (x[i] - x_1) - y[i], \\ c_{\text{lower}}[i] &= y[i] - \left(y_0 + \text{slope} \cdot (x[i] - x_0)\right), \\ c_{\text{left}}[i] &= x[i] - \min(x_0, x_0 + \frac{y_1 - y_0}{\text{slope}}), \\ c_{\text{right}}[i] &= \max(x_1, x_1 + \frac{y_1 - y_0}{\text{slope}}) - x[i]. \end{aligned}$$

Compute the constraint penalty as:

$$c_{\text{penalty}} = \min(c_{\text{upper}}, c_{\text{lower}}, c_{\text{left}}, c_{\text{right}}).$$

Step 4: Define Objective Function;

Minimize the parallelogram area while penalizing constraint violations:

$$\text{Objective: area} + \lambda \cdot \log(1 + c_{\text{penalty}}^2),$$

where:

$$\text{area} = \left|x_1 - x_0\right| \cdot \left|y_1 - y_0\right|.$$

Step 5: Perform Optimization;

Optimize x_0, x_1, and slope using the initial guess:

$$\text{result} = \text{optim}(\text{initial_guess}, \text{objective_function}, \text{method = "SANN"}).$$

Step 6: Return Optimal Parameters;

Extract and return optimal parameters:

$$x_0^{\text{opt}},\ x_1^{\text{opt}},\ \text{slope}^{\text{opt}}.$$

Algorithm 5: Add a segment to the left of the *ascending* region

Input: Time series data containing melatonin concentrations $M[i]$, timestamps $t[i]$, ascending labels $A[i]$, and threshold value τ.

Output: Updated time series with a new segment added on the left, if necessary.

Step 1: Check if All Segments Belong to the Ascending Region;

if *all points satisfy* $A[i] = 1$ **then**

Identify the leftmost node in the ascending segment:

$$i_{\text{leftmost}} = \min\{i \mid A[i] = 1\}.$$

Let the corresponding melatonin concentration and timestamp be:

$$M_{\text{leftmost}} = M[i_{\text{leftmost}}], \quad t_{\text{leftmost}} = t[i_{\text{leftmost}}].$$

end

Step 2: Check Threshold Condition for the Leftmost Node;

if $M_{leftmost} < 0.5 \cdot \tau$ **then**

Compute the melatonin concentration for the new node:

$$M_{\text{new}} = \max\left(0, \frac{M_{\text{leftmost}}}{2}\right).$$

Compute the timestamp for the new node:

$$t_{\text{new}} = t_{\text{leftmost}} - 30 \text{ minutes}.$$

end

Step 3: Add the New Segment;

Create a new node with:

$$t_{\text{new}},\ M_{\text{new}},\ A_{\text{new}} = 0,\ B_{\text{new}} = 0,\ I_{\text{new}} = 1.$$

Append the new node to the time series.

Step 4: Return Updated Time Series;

return *Time series with the added intermediate segment.*

Algorithm 6: Truncate *base* region based on threshold

Input: Time series data containing melatonin concentrations $M[i]$, timestamps $t[i]$, and base labels $B[i]$; threshold value τ (default = 2.3 pg/mL).

Output: Updated time series with truncated base segment.

Step 1: Preprocess Data;

Sort the time series data chronologically by timestamps:

$$t[1] \leq t[2] \leq \cdots \leq t[N].$$

Step 2: Identify Base Nodes Above Threshold;

For each point i in the base segment ($B[i] = 1$):

- Set $B[i] = 0$ if:

$$M[i] > \tau \quad \text{and} \quad \sum_{j=1}^{i} \mathbb{1}(B[j] = 1 \text{ and } M[j] \leq \tau) = 0.$$

- This condition ensures that all base nodes exceeding the threshold at the start of the base segment are truncated.

Step 3: Check for Remaining Base Segment;

if *all remaining points have* $B[i] = 0$ **then**

Issue a warning: "Hockey-stick time: no base part.";

return *Input data with base set to 0.*

end

Step 4: Return Updated Data;

return *Time series with truncated base segment labels* $B[i]$.

Algorithm 7: Define *intermediate* segment in a melatonin profile

Input: Time series data containing melatonin concentrations $M[i]$, timestamps $t[i]$, base labels $B[i]$, and ascending labels $A[i]$; threshold value τ.

Output: Annotated time series with intermediate segments labeled $I[i]$ (1 for intermediate, 0 otherwise).

Step 1: Preprocess Data;

Sort the time series data chronologically by timestamps;

Identify the indices of base and ascending segments:

- **Last base row:**

$$i_{\text{last_base}} = \max\{i : B[i] = 1\}.$$

- **First ascending row:**

$$i_{\text{first_asc}} = \min\{i : A[i] = 1\}.$$

Step 2: Check for Intermediate Rows;

If $i_{\text{last_base}} < i_{\text{first_asc}} - 1$:

- Identify rows between the last base and first ascending:

$$I[j] = 1 \quad \text{for } j \in (i_{\text{last_base}}, i_{\text{first_asc}}).$$

- Otherwise, set $I[j] = 0$ for all j.

Step 3: Handle Special Cases;

- **Case 1: No base points exist.**
  - If all rows before $i_{\text{first_asc}}$ have melatonin concentrations $M[j] < \min(M[i_{\text{first_asc}}], \tau)$, label those rows as intermediate:

$$I[j] = 1 \quad \text{for } j \in \{1, \ldots, i_{\text{first_asc}} - 1\}.$$

  - Additionally, set the first row as a base point:

$$B[1] = 1.$$

- **Case 2: All ascending points except the first row.**
  - If $i_{\text{first_asc}} = 2$ and $B[1] = 0$, label the first point as intermediate:

$$I[1] = 1.$$

  - Create a synthetic base point 30 minutes before the first row:

$$t_{\text{new}} = t[1] - 30 \text{ minutes}, \quad M_{\text{new}} = \frac{M[1]}{2}.$$

    Insert this new row as a base point.

Step 4: Return Annotated Data;

return *Time series with updated intermediate labels* $I[i]$.

Algorithm 8: Compute region of interest (ROI) for DLMO point search

Input: Time series data with melatonin concentrations $M[i]$, timestamps $t[i]$, base labels, ascending labels, and (if available) intermediate labels; threshold value τ (default = 2.3 pg/mL).

Output: Region of interest (ROI) with bounds: x_{start}, x_{end}, y_{min}, y_{max}.

Step 1: Preprocess Data;

Sort the time series data chronologically by timestamps;

Identify subsets of points based on labels:

- **Base points:** Points where the base label $B[i] = 1$.
- **Ascending points:** Points where the ascending label $A[i] = 1$.
- **Intermediate points (if available):** Points where the intermediate label $I[i] = 1$.

Step 2: Determine Horizontal Bounds;

- **Case 1: Intermediate points are present.**

$$x_{\text{start}} = t_{\text{last}}^{\text{base}} + 0.1 \cdot \left(t_{\text{first}}^{\text{intermediate}} - t_{\text{last}}^{\text{base}}\right),$$
$$x_{\text{end}} = t_{\text{last}}^{\text{intermediate}} + 0.95 \cdot \left(t_{\text{first}}^{\text{ascending}} - t_{\text{last}}^{\text{intermediate}}\right).$$

- **Case 2: Fewer than two base points.**

$$x_{\text{start}} = t_{\text{first}}^{\text{base}} + 0.05 \cdot \left(t_{\text{first}}^{\text{ascending}} - t_{\text{first}}^{\text{base}}\right),$$
$$x_{\text{end}} = t_{\text{last}}^{\text{base}} + 0.95 \cdot \left(t_{\text{first}}^{\text{ascending}} - t_{\text{last}}^{\text{base}}\right).$$

- **Case 3: More than two base points are available.**

$$x_{\text{start}} = \text{mean}\left(t_{\text{last-1}}^{\text{base}}, t_{\text{last}}^{\text{base}}\right),$$
$$x_{\text{end}} = t_{\text{last}}^{\text{base}} + 0.95 \cdot \left(t_{\text{first}}^{\text{ascending}} - t_{\text{last}}^{\text{base}}\right).$$

Step 3: Determine Vertical Bounds;

- The lower bound is the minimum melatonin concentration in the profile:

$$y_{\text{min}} = \min(M[i]).$$

- The upper bound is the threshold:

$$y_{\text{max}} = \tau.$$

Step 4: Return ROI;

return *ROI with bounds:* $x_{start}, x_{end}, y_{min}, y_{max}$.

Algorithm 9: Determine DLMO point using coarse and refined grid search

Input: Time series data with melatonin concentrations $M[i]$ and timestamps $t[i]$ (in POSIXct format); threshold value τ (default = 2.3 pg/mL); region of interest (ROI); grid step sizes for coarse and refined searches (Δx, Δy, Δx_{small}, Δy_{small}); fit type (e.g., linear or parabolic).

Output: Point of inflection P_{inf} (DLMO point); parameters for base and ascending segment fits.

Step 1: Generate Coarse Grid Points;

- Define the ROI as $x \in [x_{\min}, x_{\max}]$ and $y \in [y_{\min}, y_{\max}]$.
- Generate a coarse grid of candidate points:

$$G_{\text{coarse}} = \{(x, y) : x \in \text{seq}(x_{\min}, x_{\max}, \Delta x),\ y \in \text{seq}(y_{\min}, y_{\max}, \Delta y)\}.$$

Step 2: Fit Profiles and Evaluate Residuals (Coarse Grid);

foreach *candidate point* $P = (x, y) \in G_{coarse}$ **do**

- Split the data into **base segment** (points $t[i] \leq x$) and **ascending segment** (points $t[i] > x$).
- Perform linear or parabolic fitting for both segments:

$$\text{Base fit:} \quad y_{\text{base}}[i] = m_{\text{base}} \cdot (t[i] - x) + y,$$

$$\text{Ascending fit:} \quad y_{\text{asc}}[i] = \begin{cases} m_{\text{asc}} \cdot (t[i] - x) + y & \text{(linear fit)}, \\ a \cdot t[i]^2 + b \cdot t[i] + c & \text{(parabolic fit)}. \end{cases}$$

- Compute residuals:

$$R = \frac{1}{N} \sum_i \left(M[i] - y_{\text{pred}}[i]\right)^2,$$

where $y_{\text{pred}}[i]$ is the predicted melatonin value from the base or ascending fit.

if $R < R_{best}$ **then**

Update $P_{\text{best}} \leftarrow P$, $R_{\text{best}} \leftarrow R$, and store the fit parameters.

end

end

Step 3: Refine Grid Around Best Candidate;

- Identify a reduced ROI around P_{best} by expanding the range:

$$x'_{\min} = \max(x_{\text{best}} - \Delta x, x_{\min}), \quad x'_{\max} = \min(x_{\text{best}} + \Delta x, x_{\max}),$$

$$y'_{\min} = \max(y_{\text{best}} - \Delta y, y_{\min}), \quad y'_{\max} = \min(y_{\text{best}} + \Delta y, y_{\max}).$$

- Generate a refined grid of points:

$$G_{\text{refined}} = \{(x, y) : x \in \text{seq}(x'_{\min}, x'_{\max}, \Delta x_{\text{small}}),\ y \in \text{seq}(y'_{\min}, y'_{\max}, \Delta y_{\text{small}})\}.$$

Step 4: Fit Profiles and Evaluate Residuals (Refined Grid);

Repeat Step 2 over G_{refined}, further minimizing R to determine the final inflection point P_{inf}.

Step 5: Return Results;

return P_{inf} *(point of inflection), parameters for the base fit* (m_{base}) *and ascending fit* (m_{asc} *or* a, b, c)*, grid points, and residuals.*