# Package 'TimeTraits'

February 20, 2026

**Type** Package

**Title** Functional Data Analysis Pipeline, Extracting Functional Traits from Biological Time-Series Data

**Version** 1.1.0

**Description** Provides a pipeline of tools for analysing circadian time-series data using functional data analysis (FDA). The package supports smoothing of rhythmic time series, functional principle component analysis (FPCA), and extraction of group-level traits from functional representations. Analyses can incorporate multiple curve derivatives and optional temporal segmentation, enabling comparative analysis of circadian dynamics across experimental groups and time windows.

**URL** https://github.com/scllock/TimeTraits

**BugReports** https://github.com/scllock/TimeTraits/issues

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** stats, pracma, lomb, fda, fdaoutlier

**LazyData** true

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Sarah Lock [aut, cre]

**Maintainer** Sarah Lock <sarah.lock@york.ac.uk>

**Repository** CRAN

**Date/Publication** 2026-02-20 10:30:20 UTC

## Contents

---

`TimeTraits-package` *TimeTraits: Functional Data Analysis Pipeline, Extracting Functional Traits from Biological Time-Series Data.*

---

## Description

Provides tools for analysing circadian time-series using functional data analysis, including smoothing, functional principal component analysis, and extraction of group-level traits.

## Author(s)

**Maintainer**: Sarah Lock <sarah.lock@york.ac.uk>

## See Also

Useful links:

- https://github.com/scllock/TimeTraits
- Report bugs at https://github.com/scllock/TimeTraits/issues

---

`filter_curves` *Filter, detrend, and quality-control luminescence time-series data*

---

## Description

`filter_curves` preprocesses luminescence time-series data by applying a series of quality control and rhythm-detection steps. The function first subsets the data to a specified time window, removes samples with low signal intensity, detrends each time series using linear regression, and then assesses rhythmicity using Lomb–Scargle periodogram analysis. Only samples showing statistically significant rhythmic components are retained.

## Usage

```
filter_curves(
  x,
  time,
  meta = NULL,
  from = min(time, na.rm = TRUE),
  to = max(time, na.rm = TRUE),
  min_lum = 0,
  periodogram_length = NULL
)
```

## Arguments

| | |
|---|---|
| x | A numeric matrix of luminescence values, where rows correspond to time points and columns correspond to samples. Sparse matrices are allowed. |
| time | A numeric vector giving the time points corresponding to the rows of x. |
| meta | An optional data frame containing sample metadata. Must include a Sample_id column and may include a Genotype column. |
| from | Numeric value specifying the start time for analysis. Defaults to the minimum of time. |
| to | Numeric value specifying the end time for analysis. Defaults to the maximum of time. |
| min_lum | Numeric threshold specifying the minimum luminescence value required for a sample to be retained. Samples with a minimum value less than or equal to this threshold are removed. Default is 0. |
| periodogram_length | Numeric value specifying the length (in the same units as time) of the time window used for Lomb–Scargle periodogram analysis. If NULL, the full selected time range is used. |

## Details

The function returns detrended curves that pass all filtering criteria, along with optional metadata for retained samples and a log of samples removed at each step.

After subsetting the data to the requested time window, samples are filtered based on minimum luminescence to remove low-signal or failed measurements. Each remaining time series is detrended by fitting a linear model against time and extracting residuals. Rhythmicity is then assessed using a Lomb–Scargle periodogram over periods between 18 and 30 hours. Samples whose strongest spectral peak does not exceed the corresponding significance threshold are classified as non-rhythmic and removed.

A record of all removed samples and the reason for their removal is returned to support transparent quality control.

## Value

A list with the following elements:

**curves** A data frame containing the time vector and detrended luminescence curves that passed all filtering steps.

**meta_kept** A data frame of metadata corresponding to retained samples, or NULL if no metadata were supplied.

**removed** A data frame logging samples removed during filtering and the reason for removal, or NULL if no samples were removed.

## Examples

```
data(mydata_example)
data <- mydata_example[, -1]
```

```
time <- as.numeric(mydata_example[, 1])

res <- filter_curves(
  x = data,
  time = time,
  from = 24,
  to = 144,
  min_lum = 200,
  periodogram_length = 48
)
plot(res$curves[,1],res$curves[,2], xlab = "time", ylab = "relative luminescence")
```

---

`functional_traits` *Functional PCA traits for circadian rhythm data*

---

## Description

Performs functional principal component analysis (FPCA) on smoothed circadian time-series data and extracts group-level traits from the resulting FPCA score space.

## Usage

```
functional_traits(
  smooth_out,
  time,
  meta = NULL,
  group_col = "Genotype",
  groups = NULL,
  derivative = c(0, 1, 2),
  n_pc = NULL,
  segments = NULL,
  centerfns = FALSE
)
```

## Arguments

| | |
|---|---|
| `smooth_out` | An object of class `fdSmooth`, typically the output of a smoothing function applied to circadian time-series data. |
| `time` | A numeric vector giving the time points corresponding to the functional observations. |
| `meta` | An optional data frame containing sample metadata. Must include a column named `Sample_id` and a grouping column specified by `group_col`. |
| `group_col` | Character string giving the name of the column in `meta` that defines the grouping variable (e.g. genotype). Default is `"Genotype"`. |
| `groups` | Optional factor or vector defining group membership for each sample. Used only if `meta` is not supplied. Length must match the number of curves. |

| | |
|---|---|
| `derivative` | Integer vector specifying which derivatives of the functional data to analyse (e.g. 0 = original curve, 1 = first derivative). Default is `c(0, 1, 2)`. |
| `n_pc` | Integer giving the number of principal components to retain. If `NULL`, the maximum number of PCs allowed by the data is used. |
| `segments` | Optional named list defining time segments over which traits should be summarised. Each element must be a numeric vector of length two giving the start and end times (e.g. `list(pre = c(0, 104), post = c(104, 240))`). If `NULL`, the full time range is used. |
| `centerfns` | Logical indicating whether to centre functional observations before FPCA. Passed to `pca.fd`. |

## Details

The analysis can optionally be stratified by time segments (e.g. pre/post environmental shift) and by derivatives of the curves. When segments are supplied, functional observations from all segments are first re-evaluated onto a shared time grid and analysed jointly, ensuring that FPCA scores are directly comparable across segments.

For each derivative, FPCA is performed once across all samples and segments simultaneously. Segment- and group-specific traits are then computed post hoc from the shared FPCA score space.

Extracted traits include:

- Mean and standard deviation of FPCA scores (per principal component)
- Mean and standard deviation of distance to the FPCA-space centroid
- Convex hull area in PC1–PC2 space (measure of within-group diversity)

FPCA is performed in a shared functional space for each derivative. When segments are provided, all segment-specific curves are interpolated onto a common time grid prior to FPCA, ensuring statistical comparability of FPCA scores across segments. Group-level traits are computed after FPCA and do not influence the decomposition itself.

Convex hull area is computed in PC1-PC2 space only and requires at least three samples per group and segment.

## Value

A list with the following components:

**traits** A tidy data frame of group-level FPCA-derived traits, including marginal PC summaries and FPCA-space dispersion measures.

**scores** A data frame of FPCA scores for each sample, annotated with group, segment, and derivative information.

**fpca** A named list of `pca.fd` objects, one per derivative.

**meta** The metadata data frame aligned to the FPCA input order. A two-column data frame containing `Sample_id` and `Genotype`.

## See Also

`pca.fd`, `smooth_fun`, `eval.fd`

## Examples

```
time_vec <- matrix(seq(0, 48, length.out = 100))

Genotype_A <- matrix(
sin(2 * pi * (1:100) / 24) +
matrix(rnorm(100 * 5, 0, 0.1), nrow = 100, ncol = 5),
nrow = 100,
ncol = 5,
dimnames = list(NULL, paste0("A", 1:5)))

Genotype_B <- matrix(
0.3 * sin(2 * pi * (1:100) / 24) +
matrix(rnorm(100 * 5, 0, 0.5), nrow = 100, ncol = 5),
nrow = 100,
ncol = 5,
dimnames = list(NULL, paste0("B", 1:5)))

raw_data <- cbind(Genotype_A, Genotype_B)
smooth_data <- smooth_fun(raw_data, time_vec)

meta_df <- data.frame(
Sample_id = colnames(raw_data),
Genotype = rep(c("A", "B"), each = 5))

fpca_res <- functional_traits(
smooth_out = smooth_data$smooth,
time = time_vec,
meta = meta_df,
n_pc = 2)

head(fpca_res$traits)

cols <- c(A = "red", B = "blue")
with(subset(fpca_res$scores, derivative == 0),
plot(PC1, PC2, pch = 19, xlab = "PC1", ylab = "PC2", col = cols[group]))
legend("topright", legend = names(cols), col = cols, pch = 19)
```

---

# mydata_example *Arabidopsis thaliana luciferase circadian rhythm dataset*

---

## Description

Raw luminescence time-series data from an Arabidopsis thaliana luciferase reporter experiment examining circadian responses to changes in photoperiod.

## Usage

```
data(mydata_example)
```

### Format

A data frame with 300 rows and 189 variables:

**new_time** Numeric vector giving time since experiment start (hours).

**WS2.001–WS2.096** Luminescence counts per second for individual WS-2 wild-type plants. Each column corresponds to one biological replicate.

**phyb.001–phyb.096** Luminescence counts per second for individual phyB mutant plants. Each column corresponds to one biological replicate.

All luminescence values are raw photon counts per second (cps).

### Details

Plants (Ws-2 wild-type and phyB mutant lines) were entrained under short-day conditions (8 hours light: 16 hours dark) and subsequently shifted to long-day conditions (16 hours light: 8 hours dark). Luminescence was recorded at regular time intervals for approximately 166 hours.

The dataset is provided in wide format, with one column per individual plant and one row per time point (h).

### Source

Experimental data generated as part of a luciferase circadian rhythm study.

### Examples

```
data(mydata_example)
head(mydata_example)
```

---

## select_basis_lambda_LOOCV

*Select optimal number of basis functions and smoothing parameter*

---

### Description

This function performs leave-one-timepoint-out cross-validation (LOOCV) on a single functional trajectory to determine the optimal number of B-spline basis functions (‘nbasis‘) and smoothing parameter (‘lambda‘) for fitting a functional data object using `fda::smooth.basis`.

### Usage

```
select_basis_lambda_LOOCV(
  data,
  time,
  nbasis_range = seq(10, 100, 5),
  lambda_range = 10^seq(-6, 2, length.out = 10),
  norder = 4,
  penalty_order = 2
)
```

## Arguments

| | |
|---|---|
| data | A numeric vector of observed values (one curve). |
| time | A numeric vector of time points corresponding to 'data'. |
| nbasis_range | A numeric vector specifying the range of basis function numbers to test. |
| lambda_range | A numeric vector of lambda values to test (e.g. 10^seq(-6, 2, length = 10)). |
| norder | Integer specifying the order of the B-spline basis (default = 4, cubic). |
| penalty_order | Integer specifying the derivative order used in the roughness penalty (default = 2). |

## Details

Each time point is left out in turn, the curve is refit using the remaining observations, and prediction error at the omitted time point is recorded. Mean squared prediction error is used as the selection criterion.

This function is computationally expensive and intended as a helper routine for selecting smoothing parameters prior to large-scale functional analyses.

The function searches for the best combination of number of basis functions and roughness penalty by minimizing leave-one-out cross-validation error. The smoothing parameter ($\lambda$) controls the trade-off between goodness of fit and smoothness; larger values enforce smoother curves.

## Value

A list with:

| | |
|---|---|
| best_nbasis | Number of basis functions minimizing LOOCV error. |
| best_lambda | Lambda value minimizing LOOCV error. |
| cv_errors | Matrix of mean squared LOOCV errors across all combinations. |
| nbasis_range | Tested nbasis values. |
| lambda_range | Tested lambda values. |

## Examples

```
time <- seq(0, 48, length.out = 100)
y <- sin(2 * pi * time / 24) + rnorm(100, 0, 0.1)
result <- select_basis_lambda_LOOCV(y, time,
                                    nbasis_range = seq(5, 25, 5),
                                    lambda_range = 10^seq(-6, 0, length = 5))
print(result)
```

`smooth_fun` *Smooth circadian time-series data with optional shape-based outlier detection*

## Description

`smooth_fun` smooths circadian time-series data using a functional data analysis (FDA) framework based on cubic B-spline basis functions. Each curve is optionally normalised, smoothed individually, and evaluated on a common time grid. An optional shape-based outlier detection step can be applied using total variation depth, after which the remaining curves are smoothed collectively.

## Usage

```
smooth_fun(
  x,
  time,
  meta = NULL,
  normalize = TRUE,
  outlier = FALSE,
  deriv = 1,
  nbasis = 50,
  lambda = 1
)
```

## Arguments

| | |
|---|---|
| `x` | A numeric matrix of observations, where rows correspond to time points and columns correspond to samples. |
| `time` | A numeric matrix giving the time values corresponding to the rows of `x`. |
| `meta` | An optional data frame containing sample metadata. Must include a `Sample_id` column matching the column names of `x`. |
| `normalize` | Logical indicating whether each curve should be normalised to the range [0, 1] prior to smoothing. Default is `TRUE`. |
| `outlier` | Logical indicating whether shape-based outlier detection should be performed using total variation depth. If `FALSE`, all curves are retained and no outlier detection is applied. Default is `FALSE`. |
| `deriv` | Integer specifying the derivative order at which curves are evaluated for outlier detection. Default is 1 (first derivative). |
| `nbasis` | Integer specifying the maximum number of B-spline basis functions used when smoothing individual curves. The effective number of basis functions is constrained by the number of observations per curve. Default is 50. |
| `lambda` | Numeric value controlling the roughness penalty applied during smoothing of individual curves. Higher values produce smoother curves. Default is 1. |

## Details

The function supports metadata tracking, allowing identification of samples removed during outlier detection.

Each curve is first optionally normalised and stripped of missing values. Curves are smoothed individually using cubic B-spline basis functions with a second-derivative roughness penalty. Functional parameter objects are cached and reused for curves sharing the same basis configuration to improve computational efficiency.

When `outlier = TRUE`, curves are smoothed again using a higher roughness penalty and evaluated at the specified derivative order. Shape outliers are identified using total variation depth via `fdaoutlier::tvdmss` and removed prior to final smoothing.

## Value

A list with the following components:

**curves** A numeric matrix of smoothed curves evaluated on a common time grid, with columns corresponding to retained samples.

**smooth** A functional data object containing the final smoothed curves.

**time** A numeric vector giving the common time grid used for evaluation.

**meta_kept** A data frame of metadata for retained samples, or `NULL` if no metadata were supplied.

**meta_removed** A data frame of metadata for samples removed during outlier detection, or `NULL` if no samples were removed.

## Examples

```
data(mydata_example)
data <- mydata_example[, -1]
time <- as.matrix(mydata_example[, 1])

smooth_curves <- smooth_fun(
  x = data,
  time = time,
  nbasis = 40,
  lambda = 1,
  outlier = FALSE
)
plot(smooth_curves$smooth)
```

# Index