Package: TimeTraits (via r-universe)

November 14, 2026

Type Package

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

Version 1.2.0

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

Suggests tidyr, ggplot2

LazyData true

RoxygenNote 7.3.2

Config/pak/sysreqs cmake libgmp3-dev make libicu-dev libuv1-dev libssl-dev

Repository https://scllock.r-universe.dev

Date/Publication 2026-06-17 10:01:29 UTC

RemoteUrl https://github.com/scllock/timetraits

RemoteRef HEAD

RemoteSha bc839e0038e85dd19e407a3605acbe72bb2488a6

Contents

TimeTraits-package 2
filter_curves 2
functional_traits 4
metadata 8
mydata_example 8
select_basis_lambda_LOOCV 9
smooth_fun 11

Index **15**

`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,
  use_periodogram = TRUE,
  periodogram_length = NULL,
  detrend_method = "linear",
  ma_window = 24
)
```

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.
use_periodogram	Logical value specifying whether to apply Lomb–Scargle periodogram filtering to identify rhythmic samples. If TRUE, samples failing the periodogram significance threshold are removed. If FALSE, no periodogram filtering is performed. Default is TRUE.
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.
detrend_method	Character string specifying the detrending method applied prior to periodogram analysis. Options are "linear" for linear regression detrending and "moving_average" for moving-average detrending. Default is "linear".
ma_window	Numeric value specifying the window size used for moving-average detrending when detrend_method = "moving_average". Default is 24.

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

# Filter curves
filtered <- filter_curves(data,
  time = time,
  meta = metadata,
  from = 55,
  to = 153,
  min_lum = 100,
  periodogram_length = 48)
plot(filtered$curves[,1],filtered$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`.

centroid_distances A dataframe of Sample id, group, segment, derivative, PC and distance to centroid.

See Also

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

Examples

```
# Load example circadian dataset
data("mydata_example")
data("metadata")

# Filter curves
filtered <- filter_curves(mydata_example[, -1],
  time = as.matrix(mydata_example[, 1]),
  meta = metadata,
  from = 55,
  to = 153,
  min_lum = 100,
  periodogram_length = 48)

# Smooth curves
smooth_res <- smooth_fun(filtered[[1]][, -1],
  time = data.matrix(filtered[[1]][, 1]),
```

```
  meta = filtered$meta_kept,
  outlier = TRUE,
  deriv = 1,
  normalize = TRUE)

# Extract FPCA traits
fpca_res <- functional_traits(smooth_out = smooth_res$smooth,
  time       = smooth_res$time,
  meta       = smooth_res$meta_kept,
  n_pc       = 3,
  derivative = c(0, 1, 2),
  segments   = list(
  pre  = c(56, 104),
  post = c(104, 152)))

# View extracted trait table
head(fpca_res$traits)

# FPCA scores used for plotting
scores_all <- subset(fpca_res$scores,
  derivative == 1 &
    segment %in% c("pre", "post"))

# Colour groups by genotype and segment
scores_all$col <- with(scores_all,
 ifelse(segment == "pre"  & group == "WS",   "lightgreen",
       ifelse(segment == "pre"  & group == "phyb", "lightblue",
              ifelse(segment == "post" & group == "WS",   "darkgreen",
                     ifelse(segment == "post" & group == "phyb", "darkblue",
                            "grey")))))

# Plot FPCA scores
plot(scores_all$PC1,
  scores_all$PC2,
  xlab = "PC Score 1",
  ylab = "PC Score 2",
  col  = scores_all$col,
  pch  = 19)

# Add convex hulls
groups <- unique(
  interaction(scores_all$group, scores_all$segment))

for (g in groups) {
  idx <- which(interaction(scores_all$group, scores_all$segment) == g)
  if (length(idx) < 3)
    next
  h <- chull(scores_all$PC1[idx], scores_all$PC2[idx])
  polygon(scores_all$PC1[idx][h], scores_all$PC2[idx][h],
   border = unique(scores_all$col[idx]), lwd = 2)
}
```

metadata *Example genotype metadata*

Description

Metadata corresponding to the example circadian luminescence dataset included in the package(`mydata_example`).

Usage

```
data(metadata)
```

Format

A data frame with 384 rows and 2 variables:

Sample_id Sample identifier matching columns in `mydata_example`

Genotype Genotype label for each sample

Source

Internal experimental dataset

Examples

```
data(metadata)
head(metadata)
```

mydata_example *Arabidopsis thaliana luciferase circadian rhythm dataset*

Description

Luminescence time-series data from an Arabidopsis thaliana circadian reporter experiment examining transcriptional responses to photoperiod transitions.

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

The dataset contains luminescence measurements from Ws-2 wild-type seedlings harbouring the $CCR2 : LUC$ reporter construct and Ws-2 seedlings carrying the $phyB - 10$ mutation in the phytochrome B gene, also harbouring $CCR2 : LUC$.

Seeds were surface sterilised and grown on Murashige and Skoog (MS) medium supplemented with 3% sucrose before stratification for 4 days. Seedlings were entrained for 7 days under short-day conditions (8 h light : 16 h dark) at 22°C. On day 6, seedlings were transferred to black 96-well plates containing MS medium with 3% sucrose and treated with 15 μL of 5 mM D-luciferin. After one additional day of entrainment, plates were transferred to a TOPCOUNT luminometer (Perkin-Elmer), where luminescence measurements were collected approximately every 30 minutes for 6 days under blue-red light at a constant temperature of 21°C.

During the experiment, seedlings were shifted from short-day to long-day conditions (16 h light : 8 h dark) on day 11 to assess circadian responses to photoperiod change.

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

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 (λ) 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,
  n_timepoints = length(time)
)
```

Arguments

`x`	A numeric matrix of observations, where rows correspond to time points and columns correspond to samples.
`time`	A numeric vector 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.
n_timepoints	Integer specifying the number of points used in the common evaluation grid for the smoothed curves. Defaults to the number of input time points.

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.

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

Examples

```
data("mydata_example")
data("metadata")
data <- mydata_example[, -1]
time <- mydata_example[, 1]

# Filter curves
filtered <- filter_curves(data,
  time = time,
  meta = metadata,
```

```
  from = 55,
  to = 153,
  min_lum = 100,
  periodogram_length = 48)

# Smooth curves
smooth_res <- smooth_fun(filtered[[1]][, -1],
  time = data.matrix(filtered[[1]][, 1]),
  meta = filtered$meta_kept,
  nbasis = 40,
  outlier = FALSE,
  lambda = 1,
  deriv = 1,
  normalize = TRUE)

plot(smooth_res$smooth)
## Not run:
#visualise the first two removed curves
WS_keep <- as.data.frame(smooth_res$curves)
WS_removed <- as.data.frame(smooth_res$removed_curves[,1:2])

#set time and identify curve groups
time <- smooth_res$time

# add metadata columns FIRST
WS_keep$time <- time
WS_keep$curve_type <- "Normal"

WS_removed$time <- time
WS_removed$curve_type <- "Outlier"
# pivot
keep_long <- WS_keep %>%
  pivot_longer(
    cols = -c(time, curve_type),
    names_to = "sample", values_to = "value")

remove_long <- WS_removed %>%
  pivot_longer(
    cols = -c(time, curve_type),
    names_to = "sample", values_to = "value")

# combine
df_long <- rbind(keep_long, remove_long)
# plot

ggplot(df_long, aes(x = time, y = value, group = sample, color = curve_type)) +
  geom_line(data = df_long[df_long$curve_type == "Normal", ],
            alpha = 0.1, size = 1) +  # Grey lines
  geom_line(data = df_long[df_long$curve_type == "Outlier", ],
          alpha = 1, size = 1) +  # Red lines
  scale_color_manual(values = c("Normal" = "grey", "Outlier" = "red")) +
  theme_minimal() +
  theme(
```

```
    axis.text.x = element_text(size = 14, color = "black"),
    axis.title.x = element_text(size = 14, margin = margin(t = 15)),
    axis.text.y = element_text(size = 14, color = "black"),
    axis.title.y = element_text(size = 14, margin = margin(r = 15)),
    plot.margin = margin(t = 20, r = 20, b = 20, l = 20),
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12)) +
  labs(x = "Time", y = "Normalised Luminescence", color = "", title = "")
## End(Not run)
```

Index

∗ **circadian**
mydata_example, 8
∗ **datasets**
metadata, 8
∗ **functional-data-analysis**
mydata_example, 8
∗ **luminescence**
mydata_example, 8
∗ **time-series**
mydata_example, 8

eval.fd, *6*

filter_curves, 2
functional_traits, 4

metadata, 8
mydata_example, 8

pca.fd, *5*, *6*

select_basis_lambda_LOOCV, 9
smooth_fun, *6*, 11

TimeTraits *(*TimeTraits-package*)*, 2
TimeTraits-package, 2