Package 'CardioCurveR'

July 21, 2025

Title Nonlinear Modeling of R-R Interval Dynamics

Version 1.0.0

Description Automated and robust framework for analyzing R-R interval (RRi) signals using advanced nonlinear modeling and preprocessing techniques. The package implements a dual-logistic model to capture the rapid drop and subsequent recovery of RRi during exercise, as described by Castillo-Aguilar et al. (2025) <doi:10.1038/s41598-025-93654-6>. In addition, 'CardioCurveR' includes tools for filtering RRi signals using zero-phase Butterworth low-pass filtering and for cleaning ectopic beats via adaptive outlier replacement using local regression and robust statistics. These integrated methods preserve the dynamic features of RRi signals and facilitate accurate cardiovascular monitoring and clinical research.

License MIT + file LICENSE

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Imports signal (>= 1.8.1), ggplot2 (>= 3.5.1), gridExtra (>= 2.3), data.table (>= 1.16.4)

URL https://github.com/matcasti/CardioCurveR, https://matcasti.github.io/CardioCurveR/

BugReports https://github.com/matcasti/CardioCurveR/issues

Depends R (>= 3.5)

LazyData true

NeedsCompilation no

Author Matías Castillo-Aguilar [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7291-247X>)

Maintainer Matías Castillo-Aguilar <m99castillo@gmail.com>

Repository CRAN

Date/Publication 2025-04-07 16:00:09 UTC

Contents

`boot_RRi_parameters` *Bootstrap RRi Model Parameter Estimates*

Description

This function performs a bootstrap procedure on a fitted RRi model (as produced by `estimate_RRi_curve()`) to assess the uncertainty of the parameter estimates. It resamples the original data (using either a specified number of samples or a proportion of the available data) and re-estimates the dual-logistic model parameters for each bootstrap sample. The approach leverages the speed and efficiency of the **data.table** package.

Usage

```
boot_RRi_parameters(
  fit = NULL,
  n_samples = nrow(fit$data),
  prop_of_samples = NULL,
  nboot = 100
)
```

Arguments

`fit`	An object of class `"RRi_fit"` produced by `estimate_RRi_curve()`. This fitted object must contain a `data` component with the original time and RRi values as well as the fitted values.

`n_samples` A numeric value specifying the number of data points to sample for each bootstrap replicate. The default is `nrow(fit$data)` (all data points). Must not exceed the total number of rows in the data.

`prop_of_samples` A numeric value (between 0 and 1) specifying the proportion of data to use in each bootstrap sample. If specified, it overrides `n_samples` by computing the number as `floor(nrow(fit$data) * prop_of_samples)`.

`nboot` A numeric value indicating the number of bootstrap replicates to perform. The default is `100`.

Details

The bootstrap procedure returns a data.table with a row for each bootstrap replicate, containing the estimated parameters. This enables users to construct confidence intervals and assess the variability of the fitted model parameters.

The function first checks that the input `fit` object is not NULL and that `n_samples` and `nboot` are numeric and valid. If `prop_of_samples` is provided, it is used to compute the number of samples per replicate. The data from the `fit` object is converted to a data.table for efficient subsetting.

Bootstrap indices are generated by sampling with replacement, and for each bootstrap replicate, the function re-estimates the model parameters using `estimate_RRi_curve()`. The output is a data.table where each row corresponds to a bootstrap replicate.

Value

An object of class `"boot_RRi_fit"`, which is a data.table with one row per bootstrap replicate. Each row contains the estimated parameters from that bootstrap sample.

Examples

```
library(CardioCurveR)
library(data.table)

# Simulate an example RRi signal:
set.seed(123)
t <- seq(0, 20, by = 0.01)
true_params <- c(alpha = 800, beta = -350, c = 0.80,
                 lambda = -3, phi = -2, tau = 6, delta = 3)
RRi_true <- dual_logistic(t, true_params)
RRi_sim <- RRi_true + rnorm(n = length(t), sd = 30)

# Estimate the model parameters:
fit <- estimate_RRi_curve(time = t, RRi = RRi_sim)

# Bootstrap the parameter estimates using 50% of the data per replicate and 100 replicates
boot_fit <- boot_RRi_parameters(fit = fit, prop_of_samples = 0.5, nboot = 100)

# View the bootstrap estimates
print(boot_fit)
```

`CardioCurveR` *CardioCurveR: Nonlinear Modeling and Preprocessing of R-R Interval Dynamics*

Description

CardioCurveR provides an automated and robust framework for analyzing R-R interval (RRi) signals using advanced nonlinear modeling and preprocessing techniques. The package implements a dual-logistic model to capture both the rapid drop in RRi during exercise and the subsequent recovery phase, following the methodology described by Castillo-Aguilar et al. (2025):

Details

$$RRi(t) = \alpha + \frac{\beta}{1 + e^{\lambda(t-\tau)}} + \frac{-c \cdot \beta}{1 + e^{\phi(t-\tau-\delta)}}$$

In this model, α denotes the baseline RRi, β controls the amplitude of the drop, λ and τ modulate the drop phase, and c, ϕ, and δ govern the recovery dynamics.

In addition to parameter estimation, CardioCurveR offers state-of-the-art signal preprocessing tools:

CardioCurveR cleans RRi signals by applying zero-phase Butterworth low-pass filtering to remove high-frequency noise while preserving the signal phase. It further employs adaptive outlier replacement, using local regression (LOESS) and robust statistics, to identify and correct ectopic beats without "chopping" dynamic signal features.

These methods ensure that the intrinsic dynamics of RRi signals are maintained, supporting accurate cardiovascular monitoring and facilitating clinical research.

`clean_outlier` *Clean RR-Interval Signal Using Local Smoothing and Adaptive Outlier Replacement*

Description

This function cleans an RR-interval (RRi) signal by identifying ectopic or noisy beats using a robust, locally adaptive approach. In the context of cardiovascular monitoring (such as for applying the Castillo-Aguilar et al. (2025) non-linear model), the function first fits a local regression (LOESS) to the RRi signal, computes the residuals, and then identifies ectopic beats as those with residuals exceeding a multiple of the median absolute deviation.

Usage

```
clean_outlier(
  signal,
  loess_span = 0.25,
  threshold = 2,
  replace = c("gaussian", "uniform", "loess"),
  seed = 123
)
```

Arguments

signal	A numeric vector of RR interval (RRi) values.
loess_span	A numeric value controlling the span for the LOESS fit (default is 0.25). Smaller values yield a more local fit.
threshold	A numeric multiplier (default is 2) for the median absolute deviation (MAD) to determine the cutoff for flagging ectopic beats.
replace	A character string specifying the replacement method for ectopic beats. Must be one of "gaussian", "uniform", or "loess" (default is "gaussian").
seed	An integer to set the random seed for reproducibility of the replacement process (default is 123).

Details

The function offers several replacement strategies for these outliers:

"gaussian" Replace ectopic values with random draws from a normal distribution, centered at the LOESS-predicted value with a standard deviation equal to the robust MAD.

"uniform" Replace ectopic values with random draws from a uniform distribution, bounded by the LOESS-predicted value ± the MAD.

"loess" Simply replace ectopic values with the LOESS-predicted values.

This adaptive approach ensures that dynamic changes in the RRi signal, such as those observed during exercise, are preserved, while ectopic or spurious beats are corrected without "chopping" the data.

Value

A numeric vector containing the cleaned RRi signal.

Examples

```
# Simulate an RRi signal with dynamic behavior and ectopic beats:
n <- 1000

time_vec <- seq(0, 20, length.out = n)

set.seed(123)
```

```
signal <- 1000 -
  400 / (1 + exp(-3 * (time_vec - 6))) +
  300 / (1 + exp(-2 * (time_vec - 10))) + rnorm(n, sd = 50)

# Introduce ectopic beats (5% of total signal)
noise_points <- sample.int(n, floor(n * 0.05))

signal[noise_points] <- signal[noise_points] * runif(25, 0.25, 2.00)

# Clean the signal using the default Gaussian replacement strategy
clean_signal <- clean_outlier(signal = signal,
                         loess_span = 0.25, threshold = 2,
                         replace = "gaussian", seed = 123)

# Plot the signal vs cleaned signal
library(ggplot2)

ggplot() +
geom_line(aes(time_vec, signal), linewidth = 1/4, col = "purple") +
geom_line(aes(time_vec, clean_signal), linewidth = 1/4, col = "blue") +
labs(x = "Time (min)", y = "RRi (ms)",
     title = "Original (Purple) vs Cleaned signal (Blue)") +
theme_minimal()
```

`dual_logistic` *Dual-Logistic Model for RR Interval Dynamics (Castillo-Aguilar et al.)*

Description

This function implements a dual-logistic model to capture the dynamic behavior of RR intervals (RRi) during exercise and recovery, as described in Castillo-Aguilar et al. (2025). The model is designed to account for the rapid drop and subsequent recovery of RRi values by combining two logistic functions. This formulation allows for a robust characterization of the non-linear fluctuations in RRi signals, which is critical for accurate cardiovascular monitoring and analysis.

Usage

```
dual_logistic(t, params)
```

Arguments

`t`	A numeric vector of time points.
`params`	A named numeric vector of parameters, which must include:
	alpha The baseline RRi level.
	beta The amplitude parameter for the drop phase.

lambda The rate parameter controlling the steepness of the drop.

tau The time center of the drop phase.

c A scaling factor for the recovery phase.

phi The rate parameter controlling the steepness of the recovery.

delta The time offset for the recovery phase relative to `tau`.

Details

The model is defined as:

$$RRi(t) = \alpha + \frac{\beta}{1 + e^{\lambda(t-\tau)}} + \frac{-c \cdot \beta}{1 + e^{\phi(t-\tau-\delta)}}$$

where:

α is the baseline RRi level.

β controls the amplitude of the drop phase.

λ controls the steepness of the drop phase.

τ defines the time at which the drop is centered.

c scales the amplitude of the recovery phase relative to β.

ϕ controls the steepness of the recovery phase.

δ shifts the recovery phase in time relative to the drop phase.

This dual-logistic model is defined following the approach described in Castillo-Aguilar et al. (2025), and is specifically tailored for RRi signal analysis in contexts where exercise-induced changes and recovery dynamics are of interest. The model combines two logistic functions, one representing the drop in RRi and one representing the recovery, allowing for an accurate fit even in the presence of non-linear fluctuations. Attribution to Castillo-Aguilar et al. (2025) is provided to recognize the original methodology that inspired this implementation.

Value

A numeric vector containing the modeled RRi values at the times specified by `t`.

References

Castillo-Aguilar, et al. (2025). *Enhancing Cardiovascular Monitoring: A Non-linear Model for Characterizing RR Interval Fluctuations in Exercise and Recovery*. **Scientific Reports**, 15(1), 8628.

Examples

```
# Define example parameters based on Castillo-Aguilar et al. (2025)
params <- list(alpha = 1000, beta = -380, lambda = -3, tau = 6,
            c = 0.85, phi = -2, delta = 3)

# Simulate a time vector
t <- seq(0, 20, length.out = 150)
```

```
# Compute the dual-logistic model values
RRi_model <- dual_logistic(t, params)

# Plot the resulting model
library(ggplot2)

ggplot() +
geom_line(aes(t, RRi_model), linewidth = 1, col = "purple") +
  labs(x = "Time (min)", y = "RRi (ms)",
       title = "Dual-Logistic RRi Model",
       caption = "Castillo-Aguilar et al. (2025)") +
  theme_minimal()
```

`estimate_RRi_curve` *Estimate RRi Curve Using a Dual-Logistic Model for RR Interval Dynamics (Castillo-Aguilar et al.)*

Description

This function estimates parameters for a dual-logistic model applied to RR interval (RRi) signals. The model is designed to capture both the rapid drop and subsequent recovery of RRi values during exercise and recovery periods, as described in Castillo-Aguilar et al. (2025). A robust Huber loss function (with a default δ of 50 ms) is used to downweight the influence of outliers, ensuring that the optimization process is robust even in the presence of noisy or ectopic beats.

Usage

```
estimate_RRi_curve(
  time,
  RRi,
 start_params = c(alpha = 800, beta = -380, c = 0.85, lambda = -3, phi = -2, tau = 6,
    delta = 3),
  lower_lim = c(alpha = 300, beta = -750, c = 0.1, lambda = -10, phi = -10, tau =
    min(time), delta = min(time)),
  upper_lim = c(alpha = 2000, beta = -10, c = 2, lambda = -0.1, phi = -0.1, tau =
    max(time), delta = max(time)),
  method = "L-BFGS-B"
)
```

Arguments

`time`	A numeric vector of time points.
`RRi`	A numeric vector of RR interval values.
`start_params`	A named numeric vector or list of initial parameter estimates. Default is `c(alpha = 800, beta = -380, c = 0.85, lambda = -3, phi = -2, tau = 6, delta = 3)`.

`lower_lim` A named numeric vector specifying the lower bound for each parameter. Default is `c(alpha = 300, beta = -750, c = 0.1, lambda = -10, phi = -10, tau = min(time), delta = min(time))`.

`upper_lim` A named numeric vector specifying the upper bound for each parameter. Default is `c(alpha = 2000, beta = -10, c = 2.0, lambda = -0.1, phi = -0.1, tau = max(time), delta = max(time))`.

`method` A character string specifying the optimization method to use with `optim()`. The default is `"L-BFGS-B"`, which allows for parameter constraints.

Details

The dual-logistic model, as described in Castillo-Aguilar et al. (2025), is defined as:

$$RRi(t) = \alpha + \frac{\beta}{1 + e^{\lambda(t-\tau)}} + \frac{-c \cdot \beta}{1 + e^{\phi(t-\tau-\delta)}}$$

where:

α is the baseline RRi level.

β controls the amplitude of the drop.

λ modulates the steepness of the drop phase.

τ represents the time at which the drop is centered.

c scales the amplitude of the recovery relative to β.

ϕ controls the steepness of the recovery phase.

δ shifts the recovery phase in time relative to τ.

The function first removes any missing cases from the input data and then defines the dual-logistic model, which represents the dynamic behavior of RR intervals during exercise and recovery. The objective function is based on the Huber loss (with a default threshold of 50 ms), which provides robustness against outliers by penalizing large deviations less harshly than the standard squared error. This objective function quantifies the discrepancy between the observed RRi values and those predicted by the model.

Parameter optimization is performed using `optim()` with box constraints when the `"L-BFGS-B"` method is used. These constraints ensure that the parameters remain within physiologically plausible ranges. For other optimization methods, the bounds are ignored by setting the lower limit to `-Inf` and the upper limit to `Inf`.

It is important to note that the default starting parameters and bounds provided in the function are general guidelines and may not be optimal for every dataset or experimental scenario. Users are encouraged to customize the starting parameters (`start_params`) and, if necessary, the lower and upper bounds (`lower_lim` and `upper_lim`) based on the specific characteristics of their RRi signal. This customization is crucial for achieving robust convergence and accurate parameter estimates in diverse applications.

Value

A list containing:

data A data frame with columns for time, the original RRi values, and the fitted values obtained from the dual-logistic model.

method The optimization method used.

parameters The estimated parameters from the model.

objective_value The final value of the objective (Huber loss) function.

convergence An integer code indicating convergence (0 indicates success).

References

Castillo-Aguilar, et al. (2025). *Enhancing Cardiovascular Monitoring: A Non-linear Model for Characterizing RR Interval Fluctuations in Exercise and Recovery*. **Scientific Reports**, 15(1), 8628.

Examples

```
true_params <- c(alpha = 800, beta = -300, c = 0.80,
                 lambda = -3, phi = -1, tau = 6, delta = 3)

time_vec <- seq(0, 20, by = 0.01)

set.seed(1234)

# Simulate an example RRi signal:
RRi_simulated <- dual_logistic(time_vec, true_params) +
                  rnorm(length(time_vec), sd = 30)

# Estimate the model parameters:
fit <- estimate_RRi_curve(time = time_vec, RRi = RRi_simulated)

# Print method
print(fit)

# Summary method
summary(fit)

# Plot method
plot(fit)
```

`filter_signal` *Low-Pass Filter for RR Interval Signals with Edge Trimming*

Description

This function cleans an RR interval (RRi) signal by applying a Butterworth low-pass filter using zero-phase filtering (via `filtfilt` from the **signal** package) and then trimming the edges of the filtered signal to remove potential artifacts. This approach is particularly useful for preprocessing RRi data in the context of cardiovascular monitoring and non-linear modeling (see Castillo-Aguilar et al. 2025).

Usage

```
filter_signal(x, n = 3, W = 0.5, abs = 5)
```

Arguments

`x`	A numeric vector representing the RRi signal to be filtered.
`n`	An integer specifying the filter order for the Butterworth filter. Default is `3`.
`W`	A numeric value (between 0 and 1) specifying the normalized critical frequency for the low-pass filter. Default is `0.5`.
`abs`	An integer indicating the number of samples at both the beginning and end of the filtered signal to be trimmed (set to `NA`) to remove edge artifacts. Default is `5`.

Details

The filtering step is performed with a Butterworth filter of order `n` and critical frequency `W`, where `W` is normalized relative to the Nyquist frequency (i.e. a value between 0 and 1). To avoid edge artifacts produced by filtering, the function sets the first and last `abs` samples to `NA`.

This function is part of the CardioCurveR package, designed to facilitate robust analysis of RR interval fluctuations. Filtering is performed using a zero-phase forward and reverse digital filter (`filtfilt`) to ensure that the phase of the signal is preserved. The `trim` sub-function sets the first and last `abs` samples to `NA` to mitigate the impact of filter transients. These steps are crucial when preparing RRi signals for further non-linear modeling, such as the dual-logistic model described in Castillo-Aguilar et al. (2025).

Value

A numeric vector of the same length as `x` containing the denoised RRi signal, with the first and last `abs` values set to `NA`.

References

Castillo-Aguilar, et al. (2025). *Enhancing Cardiovascular Monitoring: A Non-linear Model for Characterizing RR Interval Fluctuations in Exercise and Recovery*. **Scientific Reports**, 15(1), 8628.

Examples

```
# Example: Simulate a noisy RRi signal
time <- seq(0, 60, length.out = 150)

set.seed(123)

# Simulated RRi signal (in ms) with added noise
RRi <- 1000 + sin(seq(0, 2*pi, length.out = 150)) * 50 + rnorm(150, sd = 10)

# Clean the signal using the default settings
RRi_clean <- filter_signal(x = RRi, n = 3, W = 0.5, abs = 5)

# Plot the original and filtered signals
library(ggplot2)

ggplot() +
geom_line(aes(time, RRi), linewidth = 1/2, col = "gray") +
geom_line(aes(time, RRi_clean), linewidth = 1/2, col = "purple", na.rm = TRUE) +
labs(x = "Time (s)", y = "RRi (ms)",
     title = "Original (Gray) vs Filtered Signal (Purple)") +
theme_minimal()
```

---

import_RRi_txt *Import RRi Signal from a TXT File and Preprocess It*

Description

This function imports an RR interval (RRi) signal from a plain text file, where each line contains one numeric RR interval (in milliseconds). The imported signal is preprocessed by replacing non-realistic values (those below `min` or above `max`) with `NA` and then removing them. Optionally, the function can remove ectopic beats using the `clean_outlier()` function, and it can further filter the signal using `filter_signal()`. A time variable is computed as the cumulative sum of the RR intervals (converted to minutes), and the processed data is returned as a data frame.

Usage

```
import_RRi_txt(
  file = NULL,
  remove_ectopic = TRUE,
  filter_noise = FALSE,
  min = 250,
  max = 2000,
  ...
)
```

Arguments

`file`	A character string specifying the path to the text file containing the RRi signal.
`remove_ectopic`	A logical value indicating whether to remove ectopic beats using the `clean_outlier()` function. Default is `TRUE`.
`filter_noise`	A logical value indicating whether to apply a low-pass filter using `filter_signal()` to the imported signal. Default is `FALSE`.
`min`	A numeric value specifying the minimum realistic RRi value (in milliseconds). Values below this are set to `NA`. Default is `250`.
`max`	A numeric value specifying the maximum realistic RRi value (in milliseconds). Values above this are set to `NA`. Default is `2000`.
`...`	Additional arguments passed to `readLines()`.

Details

The expected data format is a text file with one RR interval per line, for example:

[some-file.txt]:
1312
788
878
...
813
788
783

The function begins by checking that the input `file` is provided and that the options `remove_ectopic` and `filter_noise` are logical values of length 1. It then reads the file using `readLines()`, converts the readings to doubles, and replaces any values outside the realistic range (defined by `min` and `max`) with `NA`. After removing missing values, the function optionally cleans the signal to remove ectopic beats and applies a Butterworth low-pass filter if requested. Finally, it computes a time vector based on the cumulative sum of the cleaned RRi signal and returns the result in a data frame.

Value

A data frame with two columns: `time` and `RRi`. The `time` column is computed as the cumulative sum of the RRi values divided by 60000 (to convert to minutes), and `RRi` contains the cleaned signal.

Examples

```
temp_file <- tempfile(fileext = ".txt")

cat(sim_RRi$RRi_simulated,
    file = temp_file,
    sep = "\n")

sim_data <- import_RRi_txt(file = temp_file,
```

```
                          remove_ectopic = TRUE,
                          filter_noise = FALSE,
                          min = 250, max = 2000)

head(sim_data)

library(ggplot2)

ggplot(sim_data, aes(time, RRi)) +
  geom_line(linewidth = 1/4, col = "purple") +
  labs(x = "Time (min)", y = "RRi (ms)",
       title = "Processed RRi Signal") +
  theme_minimal()
```

`plot.boot_RRi_fit` *Plot method for boot_RRi_fit objects*

Description

Generates a panel of density plots to visualize the bootstrap distributions of the RRi model parameters. The method converts the bootstrap results to long format and creates one density plot per parameter.

Usage

```
## S3 method for class 'boot_RRi_fit'
plot(x, ...)
```

Arguments

`x`	An object of class "boot_RRi_fit".
`...`	Additional arguments (unused).

Value

A ggplot object with panel of density plots to visualize the bootstrap distributions of the RRi model parameters.

plot.RRi_fit *Plot method for RRi_fit objects*

Description

Produces a panel of diagnostic plots for the fitted dual-logistic model. The output includes a plot of the observed RRi signal with the fitted curve overlay, a residuals versus time plot, and a histogram of the residuals.

Usage

```
## S3 method for class 'RRi_fit'
plot(x, ...)
```

Arguments

x	An object of class "RRi_fit".
...	Additional arguments (unused).

Value

A ggplot object with a panel of diagnostic plots for the fitted dual-logistic model.

print.boot_RRi_fit *Print method for boot_RRi_fit objects*

Description

Displays a concise summary of the bootstrap RRi model parameter estimates. The printed output shows the number of bootstrap replicates and a preview of the parameter estimates from the first few replicates.

Usage

```
## S3 method for class 'boot_RRi_fit'
print(x, ...)
```

Arguments

x	An object of class "boot_RRi_fit".
...	Additional arguments passed to `print`.

Value

A message (of class `NULL`) with the number of bootstrap replicates and a preview of the first 6 bootstrap samples. Additionally, returns the input object invisibly.

print.RRi_fit *Print method for RRi_fit objects*

Description

Displays a concise summary of the RRi_fit object produced by `estimate_RRi_curve()`. The printed output includes the optimization method, estimated parameters, the final objective value, and the convergence code.

Usage

```
## S3 method for class 'RRi_fit'
print(x, ...)
```

Arguments

`x`	An object of class "RRi_fit".
`...`	Additional arguments passed to print.

Value

A message (of class `NULL`) with optimization method used, estimated parameters, objective value from the Hubber loss algorithm and the convergence code of the optimization method.

print.summary.boot_RRi_fit *Print summary of boot_RRi_fit objects*

Description

Prints a human-readable summary of the bootstrap RRi model parameter estimates, including the mean, standard deviation, and selected quantiles for each parameter.

Usage

```
## S3 method for class 'summary.boot_RRi_fit'
print(x, ...)
```

Arguments

`x`	An object of class "summary.boot_RRi_fit".
`...`	Additional arguments.

Value

A message (of class `NULL`) with bootstrapped parameter estimates, standard errors (SE) and 95% quantile-based confidence intervals.

print.summary.RRi_fit *Print summary of RRi_fit objects*

Description

Print summary of RRi_fit objects

Usage

```
## S3 method for class 'summary.RRi_fit'
print(x, ...)
```

Arguments

x	An object of class "summary.RRi_fit".
...	Additional arguments.

Value

A message (of class NULL) with optimization method used, estimated parameters, objective value from the Hubber loss algorithm, performance statistics of the RRi model and the convergence code of the optimization method.

sim_RRi *Simulated RR Interval (RRi) Data with Ectopic Beats*

Description

A data frame containing a simulated RR interval (RRi) signal generated using the dual-logistic model as described by Castillo-Aguilar et al. (2025). The data are produced by first computing a theoretical RRi curve based on specified model parameters, then adding Gaussian noise to mimic natural variability, and finally introducing ectopic beats by modifying 5% of the data points (multiplying by a factor of 0.3 or 1.7). This simulated dataset is intended for demonstrating and testing the preprocessing and modeling functions provided in the CardioCurveR package.

Usage

```
sim_RRi
```

Format

A data frame with n rows and 2 variables:

time A numeric vector of time points (in seconds).

RRi_simulated A numeric vector of simulated RR interval values (in milliseconds), including added noise and simulated ectopic beats.

Details

The dual-logistic model is defined as:

$$RRi(t) = \alpha + \frac{\beta}{1 + \exp\{\lambda(t - \tau)\}} + \frac{-c \cdot \beta}{1 + \exp\{\phi(t - \tau - \delta)\}},$$

where α is the baseline RRi level, β controls the amplitude of the drop, λ and τ define the drop phase, and c, ϕ, and δ govern the recovery.

Source

Simulated data generated using the dual-logistic model and random noise.

References

Castillo-Aguilar, et al. (2025). *Enhancing Cardiovascular Monitoring: A Non-linear Model for Characterizing RR Interval Fluctuations in Exercise and Recovery*. Scientific Reports, 15(1), 8628.

Examples

```
data(sim_RRi)

head(sim_RRi)

# Plot tha data
library(ggplot2)

ggplot(sim_RRi, aes(time, RRi_simulated)) +
geom_line(linewidth = 1/4, col = "purple") +
labs(x = "Time (s)", y = "RRi (ms)",
     title = "Simulated RRi Signal with Ectopic Beats") +
theme_minimal()
```

---

summary.boot_RRi_fit *Summary method for boot_RRi_fit objects*

Description

Computes summary statistics for each estimated parameter across the bootstrap replicates. For each parameter, the summary includes the mean, standard deviation, and the 2.5\

Usage

```
## S3 method for class 'boot_RRi_fit'
summary(object, robust = TRUE, ...)
```

Arguments

object	An object of class "boot_RRi_fit".
robust	Logical. If TRUE (default) then uses median and MAD as centrality and dispersion measures. If FALSE then uses mean and standard deviation instead.
...	Additional arguments (unused).

Value

A list with summary statistics for each parameter.

---

summary.RRi_fit *Summary method for RRi_fit objects*

Description

Provides a detailed summary of the fitted dual-logistic model, including measures of fit such as the residual sum of squares (RSS), total sum of squares (TSS), and R-squared. The summary also includes basic information about the optimization.

Usage

```
## S3 method for class 'RRi_fit'
summary(object, ...)
```

Arguments

object	An object of class "RRi_fit".
...	Additional arguments (unused).

Value

A list with the following components:

method	The optimization method used.
parameters	The estimated parameters from the model.
objective_value	The final value of the objective (Huber loss) function.
convergence	An integer code indicating convergence (0 indicates success).
RSS	Residual sum of squares.
TSS	Total sum of squares of the observed RRi values.
R_squared	Coefficient of determination.
n	The number of observations used.

Index