Package 'scanCP'

May 30, 2026

Type Package

Title Deep Learning–Based Changepoint Detection with Local Neural Models

Version 0.1.0

Description Implementation of deep learning–based changepoint detection algorithm designed for time series with smooth local fluctuations. The method fits localized feed-forward neural networks to approximate the underlying smooth component and constructs a residual-based detector that isolates abrupt structural changes. A fully data-adaptive empirical cumulative distribution function (ECDF) based thresholding rule and refinement procedures yield accurate changepoint localization without parametric assumptions on noise or trend structure.

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.3

Imports plotly, RSNNS, foreach, doSNOW, parallel, pracma, stats, magrittr, tidyr

NeedsCompilation no

Author Arman Azizyan [aut, cre], Abolfazl Safikhani [aut]

Maintainer Arman Azizyan <arman.azizyan@gmail.com>

Repository CRAN

Date/Publication 2026-05-30 13:40:08 UTC

Contents

`best_split_free` *Best Split for Binary Labels (Free Assignment)*

Description

Given a sequence of labels in {1, 2}, this function finds the best split point that maximizes the number of correctly assigned labels on each side. The left and right segments are each assigned their majority class.

Usage

```
best_split_free(labels)
```

Arguments

`labels`	Integer vector containing only 1 and 2.

Details

For each possible split point \(i\), the function computes:

- left correct = max(#1_left, #2_left)
- right correct = max(#1_right, #2_right)

and selects the split maximizing their sum.

Value

A list with:

index The optimal split location (between index and index + 1).

total_correct Maximum number of correctly assigned labels.

left_majority Majority class on the left segment.

right_majority Majority class on the right segment.

accuracy_if_assigning_majorities Proportion correctly assigned.

Examples

```
labels <- c(1,1,1,2,2,2)
best_split_free(labels)
```

`calc_detector` *Compute the MLP-Based Changepoint Detector Statistic*

Description

Computes the hybrid detector statistic used in scanCP. The detector combines a *ratio* component and a *difference* component derived from the residual sum of squares (RSS) of:

- two adjacent small-window MLP fits (window size `w`)
- one large-window MLP fit (window size `2w`)

The detector highlights locations where the large-window fit performs substantially worse than the two small-window fits, indicating a potential changepoint.

Usage

```
calc_detector(y, fit_mlp_res, w = 100,
              ma_window = w, use_abs_det = TRUE)
```

Arguments

`y`	Numeric vector. Original signal.
`fit_mlp_res`	List. Output of `fit_mlp`, containing: • `small` — list of small-window MLP predictions • `large` — list of large-window MLP predictions
`w`	Integer. Window size used for the small-window MLP.
`ma_window`	Integer. Moving-average smoothing window applied to the detector. Defaults to `w`.
`use_abs_det`	Logical. Whether to take `abs(detector)` before smoothing. Defaults to `TRUE`.

Details

For each index `i`, the detector uses:

- `rss1`: RSS of the small-window MLP on `[i, i+w-1]`
- `rss2`: RSS of the small-window MLP on `[i+w, i+2w-1]`
- `rss.tot`: RSS of the large-window MLP on `[i, i+2w-1]`

The ratio component is:

$$(rss.tot + b)/(rss1 + rss2 + b)$$

The difference component is:

$$rss.tot - rss1 - rss2$$

The final detector is a convex combination:

$$(1 - a) \cdot \text{ratio} + a \cdot \text{difference}$$

Internal parameters `a`, `b`, and `scale_01` are kept fixed at their defaults for simplicity and stability.

Value

A numeric vector of detector values of length `n - 2w`.

Examples

```
y <- rnorm(300)
fit <- fit_mlp(y, w = 100)
det <- calc_detector(y, fit, w = 100)
```

`combine_detectors` *Combine Multiple Detector Statistics into a Joint Detector*

Description

Combines several univariate detector vectors into a single joint detector using L1, L2, or max aggregation. Optionally applies moving-average smoothing and computes contribution weights.

Usage

```
combine_detectors(
  det.lst,
  method = c("L1", "L2", "max"),
  ma_window = NULL,
  circular = FALSE,
  scale_contributions = TRUE,
  scale_joint = TRUE
)
```

Arguments

det.lst	List of numeric detector vectors (all same length).
method	One of "L1", "L2", "max".
ma_window	Optional integer. If provided, each detector is smoothed using ma() with this window size.
circular	Logical. Whether smoothing wraps around.
scale_contributions	Logical. Whether to compute contribution weights.
scale_joint	Logical. Whether to scale the joint detector to [0,1].

Value

A list containing:

joint Joint detector vector.

s.mat Scaled detector matrix.

det.mat Raw detector matrix.

contrib Contribution weights (or NULL).

---

decompose_signal_core *Core Signal Decomposition (Changepoints + Piecewise Correction)*

Description

Extracts changepoints from a detector statistic using ECDF thresholding, spacing-curve analysis, and local refinement. Produces the corrected signal and its piecewise-constant component. This function does *not* perform global MLP smoothing.

Usage

```
decompose_signal_core(
  y,
  detector,
  w = 200,
  ma_window = 100,
  right_tail_cutoff = 0.95,
  left_tail_cutoff = 0.6,
  threshold = "auto",
  use_abs_det = TRUE,
  min_cp_distance = NULL,
  circular = FALSE,
  margin = NULL
)
```

Arguments

`y`	Numeric vector. Original signal.
`detector`	Numeric vector. Detector statistic produced by `calc_detector`.
`w`	Integer. Window size used in detector construction.
`ma_window`	Integer. Moving-average smoothing window for the detector and spacing curve. Defaults to `w`.
`right_tail_cutoff`	Numeric. Upper ECDF cutoff for automatic threshold selection. Defaults to `0.95`.
`left_tail_cutoff`	Numeric. Lower ECDF cutoff for automatic threshold selection. Defaults to `0.6`.
`threshold`	Either `"auto"` (default) or a numeric ECDF threshold for selecting significant peaks.
`use_abs_det`	Logical. Whether to use `abs(detector)` before peak detection. Defaults to `TRUE`.
`min_cp_distance`	Integer. Minimum distance between detected peaks. Defaults to `2 * w`.
`circular`	Logical. Whether moving-average smoothing wraps around.
`margin`	Integer. Refinement margin around each changepoint. Default `floor(w / 2)`.

Details

The function identifies local maxima of the smoothed detector, evaluates their significance using the ECDF of the detector, optionally computes a spacing curve for automatic threshold selection, and refines each changepoint using a two-cluster k-means split within a local window.

Value

A list containing:

corrected_signal The refined, piecewise-corrected signal.

piecewise_constant Estimated piecewise-constant component.

raw_correction Cumulative correction vector.

changepoints Refined changepoint locations.

changepoint_ecdf ECDF values of selected peaks.

shift_values Estimated shifts at each changepoint.

smoothed_detector Smoothed detector statistic.

ecdf_spacing Spacing-curve values (if `threshold = "auto"`).

threshold_used Final threshold applied.

local_maxima Matrix of detected local maxima.

`detect_cp_ecdf` *ECDF-Based Changepoint Detection for a Detector Statistic*

Description

Applies the ECDF + spacing-curve thresholding method to a detector vector and returns raw changepoints (no correction).

Usage

```
detect_cp_ecdf(
  diff,
  w = 200,
  ma_window = 100,
  right_tail_cutoff = 0.95,
  left_tail_cutoff = 0.6,
  threshold = "auto",
  circular = FALSE
)
```

Arguments

`diff`	Numeric detector vector.
`w`	Integer. Window size used in detector construction.
`ma_window`	Integer. Moving-average window for smoothing.
`right_tail_cutoff`	Numeric. ECDF upper cutoff.
`left_tail_cutoff`	Numeric. ECDF lower cutoff.
`threshold`	"auto" or numeric ECDF threshold.
`circular`	Logical. Whether smoothing wraps around.

Value

A list containing:

changepoints Raw changepoint indices.

ecdf_values ECDF values of selected maxima.

threshold_used Final threshold.

local_maxima Matrix of local maxima.

smoothed_detector Smoothed detector.

ecdf_spacing Spacing curve.

`fit_global_mlp` *Fit Global MLP and Compute Residual Diagnostics*

Description

Fits a global MLP model to a corrected signal and computes:

- smooth trend estimate
- residuals

This function does NOT compute a piecewise-constant component. That is handled entirely in `decompose_signal_core()`.

Usage

```
fit_global_mlp(corrected_signal)
```

Arguments

`corrected_signal`
: Numeric vector. Output of `decompose_signal_core()`.

Value

A list containing:

smooth_curve Global MLP smooth fit.

residual Residuals from the smooth fit.

mlp_model The fitted RSNNS MLP model.

`fit_mlp` *Fit Rolling MLP Models for Change Point Detection*

Description

Fits two multilayer perceptron (MLP) models over rolling windows of a univariate signal. A small-window MLP (window size `w`) captures local structure, while a large-window MLP (window size `2w`) captures broader trends. Their residual behavior is used by `calc_detector` to construct the changepoint detector.

All MLP hyperparameters are supplied through a unified `mlp_control` list for consistency with `scan_cp`.

Usage

```
fit_mlp(vec, w = 100, mlp_control = list(), parallel = FALSE)
```

Arguments

vec	Numeric vector. The input signal.
w	Integer. Window size for the small-window MLP. The large-window MLP automatically uses window size 2w.
mlp_control	A named list of MLP hyperparameters. Any subset may be supplied; unspecified values fall back to defaults. Supported fields:
	act1 Activation function for the small-window MLP ("Act_Logistic").
	act2 Activation function for the large-window MLP ("Act_Logistic").
	lr1 Learning rate for the small-window MLP (default: 0.001).
	lr2 Learning learning rate for the large-window MLP (default: 0.001).
	hl1 Number of hidden units for the small-window MLP (default: 6).
	hl2 Number of hidden units for the large-window MLP (default: 8).
	ep1 Training epochs for the small-window MLP (default: 300).
	ep2 Training epochs for the large-window MLP (default: 600).
parallel	Logical. If TRUE, rolling-window MLPs are fitted using parallel computation via **parallel**, **foreach**, and **doSNOW**. If FALSE, all computation is performed serially. Setting parallel = FALSE is recommended for CRAN checks and for systems where parallel backends are unavailable.

Details

Each rolling window is standardized before fitting. The returned matrices contain three columns:

1. standardized input x
2. fitted values y_hat
3. original indices

When parallel = TRUE, a cluster is created using parallel::makeCluster().

When parallel = FALSE, the function falls back to serial execution using %do%.

Value

A list with two elements:

small List of fitted values for each rolling window of size w.

large List of fitted values for each rolling window of size 2w.

ma *Two-Sided Moving Average Smoothing*

Description

Applies a simple two-sided moving average smoother to a numeric vector. This is used throughout the scanCP pipeline for stabilizing detector statistics and spacing curves.

Usage

```
ma(x, n = 5, circular = FALSE)
```

Arguments

x	Numeric vector to smooth.
n	Integer. Total number of neighbors used in the moving average. Must be >= 1.
circular	Logical. If TRUE, the smoothing wraps around the ends.

Details

This function is a thin wrapper around `stats::filter()` with a symmetric moving average kernel. It is intentionally lightweight and dependency-free.

Value

A numeric vector of the same length as `x` containing the smoothed values. Endpoints may be `NA` if `circular = FALSE`.

Examples

```
x <- rnorm(100)
y <- ma(x, n = 5)
```

plot_mlp_fits_interactive *Interactive Plot of Rolling MLP Fits*

Description

Creates an interactive Plotly visualization showing multiple rolling-window MLP fits over a univariate signal. A slider allows the user to move through different window positions and compare model fits dynamically.

Usage

```
plot_mlp_fits_interactive(
  y,
  fit_mlp_res,
  t.chp.ind = NA,
  step = 50,
  start = 75,
  w = 100
)
```

Arguments

y	Numeric vector. The original signal.
fit_mlp_res	A list containing rolling MLP fit results. Expected to be a list of two lists, each containing matrices with fitted values.
t.chp.ind	Optional numeric vector of changepoint indices to draw as vertical dashed lines.
step	Integer. Step size for the slider increments.
start	Integer. Starting index for the first model window.
w	Integer. Window size used in the rolling MLP fitting.

Value

A Plotly object.

---

scan_cp *Full Changepoint Detection Pipeline Using Rolling MLPs*

Description

Runs the complete changepoint detection pipeline on a univariate signal. This includes rolling-window MLP fitting, detector construction, thresholding, structural decomposition, and changepoint extraction.

Usage

```
scan_cp(
  y,
  w = 100,
  ma_window = w,
  threshold = "auto",
  threshold_tails = c(0.2, 0.95),
  min_cp_distance = 2 * w,
  margin = floor(w/2),
  use_abs_det = TRUE,
  mlp_control = list(),
  parallel = FALSE
)
```

Arguments

y	Numeric vector. The input signal.
w	Integer. Window size for the small-window MLP used in fit_mlp. The large-window MLP automatically uses window size 2w.
ma_window	Integer. Window size for the moving-average smoothing applied to the detector statistic. Defaults to w.
threshold	Character or numeric. If "auto", the threshold is selected automatically using spacing-curve analysis. Otherwise, a numeric threshold may be supplied directly.
threshold_tails	Numeric vector of length 2. Left and right tail cutoffs used when estimating the automatic threshold. Defaults to c(0.2, 0.95).
min_cp_distance	Integer. Minimum separation (in indices) between detected changepoints. Defaults to 2 * w.
margin	Integer. Margin used during local refinement of changepoint locations. Defaults to floor(w / 2).
use_abs_det	Logical. Whether to use the absolute value of the detector statistic when identifying changepoints.
mlp_control	A named list of hyperparameters passed directly to fit_mlp. Any subset may be supplied; unspecified values fall back to defaults defined in fit_mlp().
parallel	Logical. If TRUE, rolling MLP fitting is performed in parallel via fit_mlp. If FALSE, all computation is performed serially. Setting parallel = FALSE is recommended for CRAN checks and for systems without parallel support.

Details

The pipeline proceeds in three main stages:

1. Rolling-window MLP fitting via fit_mlp.
2. Detector construction via calc_detector.
3. Structural decomposition and changepoint extraction via decompose_signal_core.

Detected changepoints are printed for user convenience and returned as part of the output list. Parallel computation is optional and controlled by the parallel argument.

Value

A list containing:

changepoints Estimated changepoint locations.

detector The detector statistic computed by calc_detector.

fit_mlp Rolling MLP fit results returned by fit_mlp.

decomposition Full structural decomposition returned by decompose_signal_core.

Examples

```
# Minimal example
set.seed(1)
y <- c(rnorm(200, 0), rnorm(200, 3))


  # Full pipeline (parallel disabled for CRAN)
  res <- scan_cp(y, w = 20, parallel = FALSE)
```

`scan_cp_multi_async` *Asynchronous Multivariate Changepoint Detection*

Description

Applies the univariate `scan_cp` pipeline independently to each column of a multivariate signal. Each dimension is processed separately (asynchronously), producing a list of univariate results.

Usage

```
scan_cp_multi_async(
  Y,
  w = 100,
  ma_window = w,
  threshold = "auto",
  threshold_tails = c(0.6, 0.95),
  min_cp_distance = 2 * w,
  margin = floor(w/2),
  use_abs_det = TRUE,
  mlp_control = list()
)
```

Arguments

`Y`	Numeric matrix (n × p). Each column is a signal dimension.
`w`	Integer. Window size for rolling MLPs and detector construction. Defaults to 100.
`ma_window`	Integer. Moving-average smoothing window for the detector. Defaults to `w`.
`threshold`	Either `"auto"` (default) or a numeric ECDF threshold.
`threshold_tails`	Numeric vector of length 2 giving tail cutoff values for automatic thresholding. Defaults to `c(0.6, 0.95)`.
`min_cp_distance`	Integer. Minimum distance between detected changepoints. Defaults to `2 * w`.

margin	Integer. Local refinement margin. Defaults to floor(w/2).
use_abs_det	Logical. Whether to use abs(detector) before smoothing. Defaults to TRUE.
mlp_control	List of parameters passed to fit_mlp.

Value

A named list of length p, where each element is the output of scan_cp applied to the corresponding column of Y.

Examples

```
Y <- cbind(
  x1 = c(rnorm(200), rnorm(200, 3)),
  x2 = c(rnorm(200), rnorm(200, -2))
)

res <- scan_cp_multi_async(Y, w = 100)
res$x1$changepoints
res$x2$changepoints
```

---

scan_cp_multi_sync *Multivariate Synchronized Changepoint Detection*

Description

Implements the synchronized multivariate changepoint pipeline:

1. Fit univariate MLP-based detectors for each dimension.
2. Combine detectors into a joint detector using L1/L2/max.
3. Detect changepoints on the joint detector.
4. Compute per-dimension contributions at each detected CP.

Usage

```
scan_cp_multi_sync(
  Y,
  w = 100,
  method = c("L1", "L2", "max"),
  mlp_params = list(),
  detector_params = list(),
  combine_params = list(),
  ecdf_params = list()
)
```

Arguments

Y	Numeric matrix (n × p). Each column is a signal dimension.
w	Integer. Window size for rolling MLPs and detector.
method	Character. Combination method: "L1", "L2", or "max".
mlp_params	List of parameters passed to `fit_mlp()`.
detector_params	List of parameters passed to `calc_detector()`.
combine_params	List of parameters passed to `combine_detectors()`.
ecdf_params	List of parameters passed to `detect_cp_ecdf()`.

Value

A list containing:

detectors List of univariate detectors (one per dimension).

joint_detector Joint detector vector.

contributions Matrix of per-dimension contributions.

changepoints Detected synchronized changepoints.

cp_contributions Contribution of each dimension at each CP.

ecdf_output Full output of `detect_cp_ecdf()`.

`select_best_spike` *Select the Most Significant Spike in an ECDF Spacing Curve*

Description

Identifies the most prominent spike in a spacing curve derived from the ECDF of a smoothed detector statistic. This is used to automatically determine a significance threshold for changepoint selection.

Usage

```
select_best_spike(s,
                  right_tail_cutoff = 0.95,
                  left_tail_cutoff  = 0.6)
```

Arguments

s	Numeric vector. Smoothed spacing curve.
right_tail_cutoff	Numeric in (0,1). Exclude spikes with ECDF probability above this value (typically near 1).
left_tail_cutoff	Numeric in (0,1). Exclude spikes with ECDF probability below this value (to avoid trivial early spikes).

Details

Uses `pracma::findpeaks()` to identify local maxima. Prominence is computed as:

$$peak_{height} - \max(left_{base}, right_{base})$$

The spike with the largest prominence within the allowed ECDF range is selected.

Value

A numeric value in (0,1) representing the selected ECDF threshold, or `NA` if no valid spike is found.

Examples

```
s <- runif(100)
select_best_spike(s)
```

`simulate_piecewise_signal_idx` *Simulate a Piecewise-Constant Signal with Smooth Trend and Noise*

Description

Generate a univariate signal composed of:

- a smooth deterministic component,
- a piecewise-constant step component defined by changepoint indices,
- additive Gaussian noise.

Usage

```
simulate_piecewise_signal_idx(
  n = 1000,
  domain = c(-4, 4),
  changepoints_idx = c(300, 700),
  shift_sizes = c(0.5, -0.3, 0.7),
  noise_sd = 0.04,
  smooth_fun = function(x) 0.01 * (3 * x/2 - x^3/2),
  seed = NULL
)
```

Arguments

`n`	Integer. Length of the signal.
`domain`	Numeric vector of length 2 giving the range of the time axis.
`changepoints_idx`	Integer vector of changepoint locations (indices). Must lie strictly inside $(1, n)$.
`shift_sizes`	Numeric vector giving the mean level of each segment. Must have length equal to `length(changepoints_idx) + 1`.
`noise_sd`	Numeric. Standard deviation of the Gaussian noise.
`smooth_fun`	Function. A function $f(t)$ defining the smooth trend.
`seed`	Optional integer. If supplied, sets the random seed for reproducibility.

Details

This function is useful for benchmarking changepoint detection algorithms and reproducing controlled simulation studies.

Value

A list with components:

t Time grid of length n.

smooth Smooth component $f(t)$.

step Piecewise-constant step component.

z Final noisy signal.

changepoints_idx Sorted changepoint indices.

shift_sizes Segment means.

params List of simulation parameters.

Examples

```
sim <- simulate_piecewise_signal_idx(
  n = 1000,
  changepoints_idx = c(300, 700),
  shift_sizes = c(0.5, -0.3, 0.7),
  noise_sd = 0.05,
  seed = 123
)
```

Index