Package 'cpam'

March 4, 2026

Title Changepoint Additive Models for Time Series Omics Data

Version 0.2.0

Description Provides a comprehensive framework for time series omics analysis, integrating changepoint detection, smooth and shape-constrained trends, and uncertainty quantification. It supports gene- and transcript-level inferences, p-value aggregation for improved power, and both case-only and case-control designs. It includes an interactive 'shiny' interface. The methods are described in Yates et al. (2024) <doi:10.1101/2024.12.22.630003>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports bslib, cli, dplyr, edgeR, ggplot2, grDevices, magrittr, matrixStats, mgcv, mvnfast, parallel, pbmcapply, purrr, RColorBrewer, rlang, scam, shiny, shinyjs, stats, stringr, tidyr, tximport

Depends R (>= 3.5)

URL https://l-a-yates.github.io/cpam/, https://github.com/l-a-yates/cpam

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

BugReports https://github.com/l-a-yates/cpam/issues

VignetteBuilder knitr

NeedsCompilation no

Author Luke Yates [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1685-3169>), Michael Charleston [aut], Jazmine Humphreys [aut], Steven Smith [aut]

Maintainer Luke Yates <luke.yates@utas.edu.au>

Repository CRAN

Date/Publication 2026-03-04 03:30:02 UTC

Contents

`acat` *Aggregated Cauchy Association Test*

Description

A p-value combination method using the Cauchy distribution.

Usage

```
acat(Pvals, weights = NULL, is.check = TRUE)
```

Arguments

`Pvals`	a numeric vector/matrix of p-values. When it is a matrix, each column of p-values is combined by ACAT.
`weights`	a numeric vector/matrix of non-negative weights for the combined p-values. When it is NULL, the equal weights are used.
`is.check`	logical. Should the validity of *Pvals* (and *weights*) be checked? When the size of *Pvals* is large and one knows *Pvals* is valid, then the checking part can be skipped to save memory.

Value

The p-value(s) of acat

Author(s)

Yaowu Liu

References

Liu, Y., & Xie, J. (2019). Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures. *Journal of American Statistical Association*,115(529), 393-402. doi:10.1080/01621459.2018.1554485

Examples

```
p.values<-c(2e-02,4e-04,0.2,0.1,0.8);acat(Pvals=p.values)
acat(matrix(runif(1000),ncol=10))
```

`compute_p_values` *Compute p-values for each target ID*

Description

Compute p-values for each target ID

Usage

```
compute_p_values(
  cpo,
  subset = NULL,
  p_adj_method = "BH",
  aggregation_method = "lancaster",
  gam_method = "REML",
  gam_optimizer = "efs",
  silent = TRUE
)
```

Arguments

`cpo`	a cpam object
`subset`	a character vector of target_id names
`p_adj_method`	method for p-value adjustment
`aggregation_method`	method for aggregating target-level p-values to gene-level; either "lancaster" (default) or "acat"
`gam_method`	fitting method for `mgcv::gam` (default is "REML")
`gam_optimizer`	optimization method for `mgcv::gam` (default is "efs")
`silent`	logical; silences warnings from model fitting (default is TRUE)

Details

This function computes p-values for each target_id in the supplied cpam object. The p-values are computed from a negative binomial GAM model with a thin-plate spline basis function(s) for time using the `mgcv` package.

The p-values are stored in the new slot `p_table` in the cpam object. If `aggregate_to_gene` is set to `TRUE` (default), the target p-values are aggregated to the gene level using the `lancaster` method. The columns `p_val_target` and `p_val_gene` store the raw p-values for target- and gene-level, respectively. The function also computes adjusted p-values using the `p_adj_method`. The default method is "BH" (Benjamini-Hochberg), but any methods supported by the function `p.adjust` can be used. The adjusted p-values are stored in the columns `q_val_target` and `q_val_gene`.

Value

an updated cpam object with raw, adjusted, and possibly aggregated p-values stored in the new slot "p_table"

References

Wood, S.N. (2013a) On p-values for smooth components of an extended generalized additive model. Biometrika 100:221-228 doi:10.1093/biomet/ass048

Yi L, Pachter L (2018). aggregation: p-Value Aggregation Methods. R package version 1.0.1, https://CRAN.R-project.org/package=aggregation.

Examples

```
library(cpam)

# load gene-only example cpam object
load(system.file("extdata", "cpo_example.rda", package = "cpam"))

# run on a small subset of the example data
cpo <- compute_p_values(cpo_example, subset = paste0("g00",1:9))
cpo$p_table
```

---

cpam-class *The cpam class*

Description

The `cpam` class stores data and analysis results for a time series omics experiment. This object is generated by the `prepare_cpam` function and contains all necessary data and parameters for downstream analysis.

Details

A cpam object is a list with the following components:

exp_design A data frame containing the experimental design information, with columns for 'sample', 'time', and potentially other variables.

count_matrix_raw The original count matrix before filtering.

count_matrix_filtered The count matrix after filtering low-count genes/transcripts.

target_to_keep A vector of transcript/gene IDs that passed the filtering criteria.

data_long A long-format data frame containing all relevant information for each target and sample.

t2g A transcript-to-gene mapping data frame if provided.

regularize Logical; whether empirical Bayes regularization of dispersions was used.

overdispersion.prior Median overdispersion.

model_type String; the type of design used, either "case-only" or "case-control".

condition_var String; the column name in exp_design for the condition variable (for case-control models).

case_value The value of condition_var that indicates the "case" in case-control models.

bootstrap Logical; whether bootstrap samples (inferential replicates) were used.

nboot The number of bootstrap samples used, if applicable.

filter A list containing the filtering function and its arguments used.

gene_level Logical; whether the analysis was performed at the gene level.

aggregate_to_gene Logical; whether p-values should be aggregated from transcript to gene level.

times An ordered vector of unique time points in the experimental design.

num_cores The number of cores used for parallel computation.

fixed_effects The formula for fixed effects in the model.

intercept_cc String; the intercept type for case-control models.

bss A vector of basis function types used for modelling.

Methods

Objects of class cpam have print and summary methods available.

See Also

prepare_cpam for creating a cpam object.

Examples

```
# load gene-only example data
load(system.file("extdata", "exp_design_example.rda", package = "cpam"))
load(system.file("extdata", "count_matrix_example.rda", package = "cpam"))

# Create a cpam object with the example data
cpo <- prepare_cpam(exp_design = exp_design_example,
```

```
                    count_matrix = count_matrix_example,
                    gene_level = TRUE)

# Print the object structure
cpo

# Get a summary of the cpam object
summary(cpo)
```

`estimate_changepoint` *Use model selection to estimate changepoints*

Description

Use model selection to estimate changepoints

Usage

```
estimate_changepoint(
  cpo,
  cps = NULL,
  degs_only = TRUE,
  deg_threshold = 0.05,
  subset = NULL,
  sp = NULL,
  bss = "tp",
  family = c("nb", "gaussian"),
  score = "aic",
  compute_mvn = FALSE
)
```

Arguments

`cpo`	a cpam object
`cps`	vector of candidate changepoints. Defaults to the set of observed timepoints
`degs_only`	logical; should changepoints only be estimated for differentially expressed genes
`deg_threshold`	logical; the threshold value for DEGs (ignored of `degs_only = F`)
`subset`	character vector; names of targets or genes (if `cpo$gene_level = T`) for which changepoints will be estimated
`sp`	numerical >= 0; supply a fixed smoothing parameter. This can decrease the fitting time but it is not recommended as changepoints estimation is sensitive to smoothness.
`bss`	character vector; names of candidate spline bases (i.e., candidate shape types). Default is thin plate ("tp") splines.

family	character; negative binomial ("nb", default) or Gaussian ("gaussian", not currently supported)
score	character; model selection score, either Generalised Cross Validation ("gcv") or Akaike Information Criterion ("aic")
compute_mvn	logical; use simulation to compute p-value under multivariate normal model of the model scores (default FALSE)

Details

This function estimates changepoints for each target_id. The assumed trajectory type for this modelling stage is initially constant followed by a changepoint into thin-plate smoothing spline.

By default, candidate time points are limited to the discrete observed values in the series, since, despite the use of smoothing constraints, there is generally insufficient information to infer the timing of changepoints beyond the temporal resolution of the data. In any case, the candidate points can be set manually using the cps argument.

To estimate changepoints, a model is fit for each candidate changepoint and generalised cross-validation (GCV, default) or the Akaike Information Criterion (AIC) are used to select among them. Model-selection uncertainty is dealt with by computing the one-standard-error rule, which identifies the least complex model within one standard error of the best scoring model.

Both the minimum and the one-standard-error (default) models are stored in the returned slot "changepoints" so that either can be used. Optionally (if compute_mvn = TRUE), this function also computes the probability (denoted p_mvn) that the null model is the best scoring model, using a simulation-based approach based on the multivariate normal model of the pointwise model scores.

Given the computational cost of fitting a separate model for each candidate changepoint, cpam only estimates changepoints for targets associated with 'significant' genes at the chosen threshold deg_threshold.

Value

a cpam object with the estimated changepoint table added to the slot "changepoints"

References

Yates, L. A., S. A. Richards, and B. W. Brook. 2021. Parsimonious model selection using information theory: a modified selection rule. Ecology 102(10):e03475. 10.1002/ecy.3475

Examples

```
library(cpam)
library(dplyr)

# load example data
load(system.file("extdata", "exp_design_example.rda", package = "cpam"))
load(system.file("extdata", "count_matrix_example.rda", package = "cpam"))

cpo <- prepare_cpam(exp_design = exp_design_example,
                    count_matrix = count_matrix_example[1:20,],
                    gene_level = TRUE,
```

```
                    num_cores = 1)
cpo <- compute_p_values(cpo)
cpo <- estimate_changepoint(cpo)
cpo$changepoints
```

`plot_cluster` *Plot clustered targets*

Description

Plot clustered targets

Usage

```
plot_cluster(cpo, res, changepoints, shapes, alpha = 0.1)
```

Arguments

`cpo`	a cpam object
`res`	a tibble, output from `results()` containing columns target_id, cp, and shape
`changepoints`	numerical or character; one or more changepoints (these should be the same as the ones used in `estimate_changepoint()`
`shapes`	character; one or more shapes (these should be the same as the ones used in `select_shape()`
`alpha`	numeric between 0 and 1; controls line transparency in plot (default: 0.1)

Details

Plots the fitted trends for a set of targets whose estimated changepoints and shapes are given by the arguments `changepoints` and `shapes`, respectively.

Creates a combined plot showing fitted expression trends for all targets that share specified changepoint times and shape patterns. Each line represents one target's fitted trajectory, with transparency controlled by alpha.

Value

A ggplot object showing overlaid fitted trends, or NULL if no matching targets are found

See Also

`results()`, `plot_cpam()`

Examples

```
library(cpam)

# load gene-only example cpam object
load(system.file("extdata", "cpo_example.rda", package = "cpam"))

# Generate results table
res_example <- results(cpo_example)

# plot all targets with changepoint at timepoint 0 and shape "ilin" (increasing linear)
plot_cluster(cpo_example, res_example, changepoints = 2, shapes = "ilin")
```

---

plot_cpam *Plot fitted changepoint additive models*

Description

Plot fitted changepoint additive models

Usage

```
plot_cpam(
  cpo,
  gene_id = NULL,
  target_id = NULL,
  cp_type = c("cp_1se", "cp_min"),
  shape_type = "shape1",
  bs = "auto",
  cp_fix = -1,
  facet = FALSE,
  sp = NULL,
  show_fit = TRUE,
  show_data = TRUE,
  show_fit_ci = TRUE,
  show_data_ci = TRUE,
  ci_prob = "se",
  remove_null = FALSE,
  null_threshold = 0.05,
  null_threshold_adj = TRUE,
  k_mult = 1.2,
  return_fits_only = FALSE,
  family = "nb",
  common_y_scale = TRUE,
  scaled = FALSE,
  base_size = 12
)
```

Arguments

cpo	A cpam object containing count data, model fits, and optional changepoint/shape estimates
gene_id	character; gene_id (mutually exclusive with target_id)
target_id	character; target_id (mutually exclusive with gene_id)
cp_type	One of "cp_1se" or "cp_min"; rule for selecting changepoint from fitted models. See `estimate_changepoint()` for details.
shape_type	One of "shape1" or "shape2"; which set of fitted shape patterns to use. See `select_shape()` for details.
bs	Shape pattern to fit ("null", "lin", "ilin", "dlin", or from cpo$bss). Use "auto" (default) to use estimated shapes as per `shape_type`.
cp_fix	Numeric; fixed changepoint time. Set to -1 (default) to use estimated changepoints
facet	Logical; for multiple transcripts, plot in separate facets?
sp	numerical; set the smooth parameter. NULL (default) for automatic selection
show_fit	logical; show the fitted trend?
show_data	logical; show (possibly normalized and scaled) data points?
show_fit_ci	logical; show credible interval for the fitted trend?
show_data_ci	logical; show bootstrapped quantile for data points?
ci_prob	"se" for standard error bands (see `mgcv::predict.gam()`), or numeric for simulation-based intervals. If numerical, sets the probability for the simulation-based estimates of credible interval.
remove_null	logical; only plot differentially expressed transcripts (not applicable for gene-only analyses)
null_threshold	numeric; P value threshold for filtering out NULL transcripts
null_threshold_adj	logical; use adjusted (default) or non-adjusted p-values for filtering targets
k_mult	numerical; multiplier for the number of knots in the spline. Not recommended to change this value.
return_fits_only	logical; return the model fits. Does not plot the function
family	character; negative binomial ("nb", default) or Gaussian ("gaussian")
common_y_scale	logical; for faceted plots of multiple transcripts, should the scale of the y-axis be common or free.
scaled	logical; scaled data by overdispersions (for bootstrapped data only)
base_size	numeric; base font size for the plot

Details

Plots the fitted trend and data points for a given gene or target. If a gene ID is supplied, the function will plot all transcripts for that gene. The function can also be used to return the model fit(s) only, which are `gamObject` objects from the `mgcv` package.

Value

a ggplot object

Examples

```
library(cpam)

# load gene-only example cpam object
load(system.file("extdata", "cpo_example.rda", package = "cpam"))

# example gene
plot_cpam(cpo_example, gene_id = "g003")

# gene with estimated changepoint at timepoint 3
plot_cpam(cpo_example, gene_id = "g013")

# manually set the changepoint
plot_cpam(cpo_example, gene_id = "g013", cp_fix = 2)
```

---

prepare_cpam *Prepare a cpam object*

Description

Prepare a cpam object

Usage

```
prepare_cpam(
  exp_design,
  count_matrix = NULL,
  t2g = NULL,
  import_type = NULL,
  model_type = c("case-only", "case-control"),
  bootstrap = TRUE,
  filter_fun = "ts_filter",
  filter_fun_args = list(min_reads = 5, min_prop = 3/5),
  regularize = TRUE,
  gene_level = FALSE,
  aggregate_to_gene = !gene_level,
  condition_var = "condition",
  case_value = "treatment",
  num_cores = 1,
  normalize = TRUE,
  fixed_effects = NULL,
  intercept_cc = c("1", condition_var)
)
```

Arguments

exp_design	a dataframe or tibble with the experimental design, containing at least a 'time' and a 'sample' column
count_matrix	a matrix of counts. Column names must be in 'sample' column of exp_design,
t2g	a transcript to gene dataframe or tibble with columns target_id and gene_id
import_type	software used for quantification, one of "kallisto", "salmon" ,.... Ignored if count_matrix is supplied.
model_type	"case-only" (default) or "case-control"
bootstrap	logical; load bootstrap samples, also called inferential replicates, if available, and rescale counts.
filter_fun	filter function to remove lowly expressed genes (default is filter_fun())
filter_fun_args	arguments for filter function
regularize	logical; use empirical Bayes regularization of dispersions (default is TRUE)
gene_level	logical; aggregate counts to gene level before data preparation and modelling (default is FALSE)
aggregate_to_gene	logical; aggregate p values from transcript- to gene-level
condition_var	string; column name in exp_design for the condition variable (for model_type = "case_control" only)
case_value	value of condition_var that indicates the "case". All other values are deemed to be control
num_cores	integer; number of cores to use for parallel computation
normalize	logical; use model offsets based on sampling depth and gene length
fixed_effects	a model formula of the form ~ effect1 + effect2
intercept_cc	string; intercept for case-control model: "1" (default) for common intercept or "condition"

Details

This function prepares a cpam object for analysis. The function loads count data from files or a matrix, filters lowly expressed genes, computes normalisation factors, and estimates dispersions. Many of these steps can be customised or turned off.

When bootstrap samples (inferential replicates) are available, it loads and summarises these using means, standard errors, and estimated overdispersions. The latter are a measure of quantification uncertainty and they are used to rescale the counts which accounts for this uncertainty during the modelling steps.

Value

an object of class cpam-class. The returned object has methods print and summary for displaying information. See cpam-class for details on the structure of the returned object.

References

Pedro L Baldoni, Yunshun Chen, Soroor Hediyeh-zadeh, Yang Liao, Xueyi Dong, Matthew E Ritchie, Wei Shi, Gordon K Smyth, Dividing out quantification uncertainty allows efficient assessment of differential transcript expression with edgeR, Nucleic Acids Research, Volume 52, Issue 3, 9 February 2024, Page e13, https://doi.org/10.1093/nar/gkad1167

Yunshun Chen, Lizhong Chen, Aaron T L Lun, Pedro L Baldoni, Gordon K Smyth, edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets, Nucleic Acids Research, Volume 53, Issue 2, 27 January 2025, https://doi.org/10.1093/nar/gkaf018

Examples

```
library(cpam)

# load gene-only example data
load(system.file("extdata", "exp_design_example.rda", package = "cpam"))
load(system.file("extdata", "count_matrix_example.rda", package = "cpam"))

cpo <- prepare_cpam(exp_design = exp_design_example,
                    count_matrix = count_matrix_example,
                    gene_level = TRUE)
cpo
```

---

results *Create a results table from a cpam object*

Description

Create a results table from a cpam object

Usage

```
results(
  cpo,
  p_threshold = 0.05,
  p_type = c("p_gam", "p_mvn"),
  min_lfc = 0,
  min_count = 0,
  aggregate_to_gene = cpo$aggregate_to_gene,
  add_lfc = TRUE,
  add_counts = TRUE,
  cp_type = c("cp_1se", "cp_min"),
  shape_type = c("shape1", "shape2"),
  summarise_to_gene = FALSE,
  remove_null_targets = TRUE
)
```

Arguments

cpo	a cpam object
p_threshold	numerical; threshold for adjusted p-values; default is 0.05
p_type	character; choose the type of p-value. Options are "p_gam" (default) or "p_mvn" (see compute_p_values() for details).
min_lfc	numerical; maximum absolute log (base 2) fold change must exceed this minimum value; default is 0
min_count	numerical; maximum of the modelled counts evaluated at the set of observed time points must exceed this minimum value for
aggregate_to_gene	logical; filter by gene-aggregated p-values
add_lfc	logical; add log (base 2) fold changes for each time point
add_counts	logical; add modelled counts for each time point
cp_type	character; model-selection rule used to select the changepoint
shape_type	character; "shape1" to include unconstrained or otherwise "shape2"
summarise_to_gene	logical; return gene-level results only
remove_null_targets	logical; remove targets with null shapes (default is T). If F, targets with null shapes will be included if the aggregated p-value for the corresponding gene passes the specified filtering thresholds.

Details

This function is usually called after compute_p_values(), estimate_changepoint, and select_shape have been run. The function has several useful filters such as adjusted p-value thresholds, minimum log-fold changes, and minimum counts.

Value

a tibble

Examples

```
library(cpam)

# load gene-only example cpam object
load(system.file("extdata", "cpo_example.rda", package = "cpam"))

results(cpo_example)

# Add filters
results(cpo_example, p_threshold = 0.01, min_lfc = 1)
```

`select_shape` *Use model selection to select a shape for each target*

Description

Use model selection to select a shape for each target

Usage

```
select_shape(
  cpo,
  subset = NULL,
  sp = NULL,
  bss = c("micv", "mdcx", "cv", "cx", "lin", "tp", "null"),
  family = c("nb", "gaussian"),
  score = "gcv",
  cp_type = c("cp_1se", "cp_min")
)
```

Arguments

`cpo`	a cpam object
`subset`	character vector; names of targets or genes (if `cpo$gene_level = T`) for which changepoints will be estimated
`sp`	numerical >= 0; supply a fixed smoothing parameter. If NULL (default), the smoothing parameter is estimated. Note, this fixed value is in any case applied only to shape constrained bases (i.e., not `bs = 'tp'`).
`bss`	character vector; names of candidate spline bases (i.e., candidate shape types).
`family`	character; negative binomial ("nb", default) or Gaussian ("gaussian")
`score`	character; model selection score, either Generalised Cross Validation ("gcv") or Akaike Information Criterion ("aic")
`cp_type`	character; if changepoints have been estimated using `estimate_changepoint()`, which selection rule should be used. See `estimate_changepoint()` for details.

Details

The function selects the best shape from a list of candidate shapes for each target. It is typically the last step in the analysis, called after p-values have been estimated using `compute_p_values()` and changepoints have been estimated using `estimate_changepoint()`.

Two shape selections are generated. The first selecting among linear, convex and concave shape classes and their monotonic variants (or the shape set given by bss), and the second selecting among the first options plus an 'unconstrained' smooth. The inclusion of the 'unconstrained' type provides the flexibility to detect targets beyond simpler trends. For computational reasons, as per the changepoint estimation, shapes are selected only for those genes, or their isoforms, identified as significant at the chosen FDR threshold. This is overridden by providing a subset of target names to the `subset` argument, provided these targets have estimated changepoints.

Value

a cpam object with the selected shapes added to the slot "shapes"

Examples

```
library(cpam)

# load example data
load(system.file("extdata", "exp_design_example.rda", package = "cpam"))
load(system.file("extdata", "count_matrix_example.rda", package = "cpam"))

# Using a small subset of the example data
cpo <- prepare_cpam(exp_design = exp_design_example,
                    count_matrix = count_matrix_example[1:20,],
                    gene_level = TRUE,
                    num_cores = 1)
cpo <- compute_p_values(cpo)
cpo <- estimate_changepoint(cpo)
cpo <- select_shape(cpo)
cpo$shapes
```

`ts_filter` *Removes lowly expressed genes*

Description

Removes lowly expressed genes

Usage

```
ts_filter(data, min_reads = 5, min_prop = 3/5)
```

Arguments

`data`	A tibble or data.frame containing columns: • target_id (character): Transcript identifiers • time (numeric): Time point of measurement • counts (numeric): Read counts
`min_reads`	minimum reads per transcript per sample
`min_prop`	minimum proportion of samples that exceed `min_read` at a given time point (default: 3/5)

Details

Identifies targets that show strong and consistent expression in at least one timepoint. For each timepoint, the function calculates the proportion of samples where a targets exceeds `min_reads`. Targets are retained if they meet the minimum proportion (`min_prop`) at any timepoint in the experiment.

Value

a character vector of transcript IDs to keep

Examples

```
data <- dplyr::tibble(
  target_id = rep(paste0("t", 1:3), each = 6),
  time = rep(c(0, 4, 8), 6),
  counts = c(6,6,6, 0,0,0, 6,0,6, 0,6,6, 6,6,6, 6,0,0)
)
ts_filter(data)
```

`visualise` *Launches a Shiny app to visualise the data and fitted models of a cpam object*

Description

Launches a Shiny app to visualise the data and fitted models of a cpam object

Usage

```
visualise(
  cpo,
  subset = NULL,
  degs_only = TRUE,
  deg_threshold = 0.05,
  p_type = c("p_gam", "p_mvn"),
  shape_type = c("shape1", "shape2")
)
```

Arguments

`cpo`	A cpam object containing count data, model fits, and optional changepoint/shape estimates
`subset`	Character vector; names of targets or genes (if `cpo$gene_level = TRUE`) to load into the Shiny app. If NULL, all genes/targets are included based on `degs_only`.
`degs_only`	Logical; if TRUE, display only differentially expressed genes/targets with adjusted p-value below `deg_threshold`. Default is TRUE.
`deg_threshold`	Numeric; significance threshold for differentially expressed genes/targets. Only used when `degs_only = TRUE`. Default is 0.05.
`p_type`	character; choose the type of p-value. Options are "p_gam" (default) or "p_mvn" (see `compute_p_values()` for details).

shape_type — character; "shape1" to include unconstrained or otherwise "shape2". Default is "shape1". In some instances, all of the transcripts for a gene may be "null" shaped, but the p-value for the gene may still be significant. This is due to the different methods of determining significance for the changepoints and the gene-level p-values. Here, conservatively, we remove these null-shaped genes from the DEG list.

Value

None (launches Shiny app in browser)

Examples

```
if (interactive()){

# A simple gene-level example
cpo <- cpo_example

# Launch visualization with all genes
visualise(cpo, degs_only = FALSE)

# Launch with only significant genes
visualise(cpo, deg_threshold = 0.05)

# Launch with specific genes
visualise(cpo, subset = c("g001","g002","g003"))
}
```

Index