Package 'OptSurvCutR'

June 30, 2026

Type Package

Title Optimal Survival Cut-Point Discovery for Time-to-Event Analysis with 'OptSurvCutR'

Version 0.10.0

Description Provides a robust workflow for optimal cut-point analysis in time-to-event ('survival') data. Functions determine the optimal number of cut-points via find_cutpoint_number(), find their precise locations via find_cutpoint() using systematic or genetic algorithms (via the 'rgenoud' package), and validate stability via bootstrapping using validate_cutpoint(). Features include covariate adjustment, parallel processing, and an extensible S3 plotting engine for clinical dashboards and diagnostics.

License GPL-3

URL https://github.com/paytonyau/OptSurvCutR, https://paytonyau.github.io/OptSurvCutR/

BugReports https://github.com/paytonyau/OptSurvCutR/issues

Encoding UTF-8

VignetteBuilder knitr

Imports cli, doParallel, doRNG, foreach, ggplot2, parallel, patchwork, Rcpp, rgenoud, rlang, stats, survival, survminer, tidyr

LinkingTo Rcpp

Suggests broom, dplyr, knitr, plotly, readxl, rmarkdown, spelling, testthat (>= 3.0.0), timeROC

Config/testthat/edition 3

Depends R (>= 3.5)

Language en-GB

Config/Needs/website logo.png

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Payton Yau [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3283-0370>)

Maintainer Payton Yau <tungon@gmail.com>

Repository CRAN

Date/Publication 2026-06-30 12:20:26 UTC

Contents

find_cutpoint 2
find_cutpoint_number 5
plot.find_cutpoint 8
plot_cutpoint_residuals 9
plot_landmark_stratification 10
plot_optimisation_curve 11
plot_validation 12
theme_optsurv 13
validate_cutpoint 14

Index **16**

`find_cutpoint` *Find Optimal Cut-points for Survival Data*

Description

Finds optimal cut-point(s) for a continuous predictor in a time-to-event (survival) analysis. Uses systematic search (1–2 cuts) or a genetic algorithm (any number of cuts). Features high-speed integer partitioning via compiled C++ vector assignments and automated quantile grid downsampling.

Usage

```
find_cutpoint(
  data,
  predictor,
  outcome_time,
  outcome_event,
  num_cuts = 1,
  method = c("systematic", "genetic"),
  criterion = c("logrank", "hazard_ratio", "p_value"),
  covariates = NULL,
  nmin = 20,
  seed = NULL,
  max.generations = NULL,
  pop.size = NULL,
  n_perm = 0,
  n_cores = 1,
```

```
  use_cpp = TRUE,
  grid_by = 0.01,
  quiet = FALSE,
  candidate_cuts = NULL,
  ...
)

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

## S3 method for class 'find_cutpoint'
summary(
  object,
  show_model = TRUE,
  show_group_counts = TRUE,
  show_medians = TRUE,
  show_ph_test = TRUE,
  show_params = TRUE,
  ...
)
```

Arguments

data	A data frame containing the analysis variables.
predictor	The continuous predictor variable name (character).
outcome_time	The time-to-event variable name (character).
outcome_event	The event status variable name (character, 0 or 1).
num_cuts	The number of cut-points to find. Default is 1.
method	Algorithm search type: ‘"systematic"‘ or ‘"genetic"‘.
criterion	The statistic to optimise: ‘"logrank"‘, ‘"hazard_ratio"‘, or ‘"p_value"‘.
covariates	Character vector of covariate names (optional).
nmin	Min. group size (integer count or proportion).
seed	Optional integer seed for reproducible genetic search.
max.generations	Max generations for genetic algorithm. If ‘NULL‘, dynamically scales.
pop.size	Population size for genetic algorithm. If ‘NULL‘, dynamically scales.
n_perm	Number of permutations to run for an adjusted p-value. Default is 0.
n_cores	Number of CPU cores for parallel permutations. Default is 1.
use_cpp	Logical. Checks and calls compiled C++ routines via ‘Rcpp‘. Default is ‘TRUE‘.
grid_by	Percentile step increment for systematic grid downsampling. Default is 0.01.
quiet	Logical. If ‘TRUE‘, suppresses operational console alerts.
candidate_cuts	Optional vector of pre-filtered cuts defining a narrow search space.
...	Additional arguments passed down to downstream rendering pipelines.

x	A find_cutpoint result object.
object	A find_cutpoint result object for summary evaluation.
show_model	Logical. Whether to print the full Cox model summary frame.
show_group_counts	Logical. Whether to show stratified sample split counts.
show_medians	Logical. Whether to display Kaplan-Meier median tracking times.
show_ph_test	Logical. Display the proportional hazards validation check.
show_params	Logical. Print original baseline parameters.

Details

'method = "systematic"': grid search respecting 'nmin'. Optimised via internal quantiles. 'method = "genetic"': 'rgenoud' global optimisation. Systematic search is slow for 'num_cuts > 2'; use 'genetic'. Core vector partitions are calculated in compiled C++ via 'Rcpp' for optimal performance.

Value

An object of class 'find_cutpoint' containing the optimal cut-points, statistic, and analysis parameters.

srrstats compliance

.

.

.

References

Altman, D. G., Lausen, B., Sauerbrei, W., & Schumacher, M. (1994). Dangers of Using "Optimal" Cutpoints in the Evaluation of Prognostic Factors. *JNCI: Journal of the National Cancer Institute*, 86(11), 829–835. doi:10.1093/jnci/86.11.829

Cox, D. R. (1972). Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2), 187–202. doi:10.1111/j.25176161.1972.tb00899.x

Mantel, N. (1966). Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemotherapy Reports*, 50(3).

Mebane Jr, W. R., & Sekhon, J. S. (2011). Genetic Optimisation Using Derivatives: The rgenoud Package for R. *Journal of Statistical Software*, 42, 1–26. doi:10.18637/jss.v042.i11

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)

  # Create a lightweight, reproducible simulation baseline dataset
  set.seed(42)
  sim_data <- data.frame(
    time = rexp(30, rate = 0.1),
```

```
  event = sample(c(0, 1), 30, replace = TRUE),
  biomarker = rnorm(30, mean = 5, sd = 1.5)
)

# Execute an exhaustive systematic threshold discovery sweep
fit <- find_cutpoint(
  data = sim_data,
  predictor = "biomarker",
  outcome_time = "time",
  outcome_event = "event",
  num_cuts = 1,
  method = "systematic",
  criterion = "logrank",
  nmin = 5,
  quiet = TRUE
)
print(fit)
}
```

`find_cutpoint_number` *Find Optimal Number of Cut-points for Survival Data*

Description

Finds optimal cut-point number (0 to ‘max_cuts‘) for a Cox model by comparing AIC, AICc, or BIC. Features hardware-accelerated grouping iterations via Rcpp compilation hooks and robust UX constraint warnings.

Usage

```
find_cutpoint_number(
  data,
  predictor,
  outcome_time,
  outcome_event,
  method = "systematic",
  criterion = "BIC",
  covariates = NULL,
  max_cuts = 2,
  nmin = 0.1,
  seed = NULL,
  max.generations = NULL,
  pop.size = NULL,
  use_cpp = TRUE,
  ...
)

## S3 method for class 'find_cutpoint_number_result'
```

```
print(x, ...)

## S3 method for class 'find_cutpoint_number_result'
summary(
  object,
  show_comparison_table = TRUE,
  show_best_model_details = TRUE,
  show_group_counts = TRUE,
  show_medians = TRUE,
  show_ph_test = TRUE,
  plot.it = FALSE,
  ...
)

## S3 method for class 'find_cutpoint_number_result'
plot(x, y, ...)

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

## S3 method for class 'find_cutpoint_number_result'
summary(
  object,
  show_comparison_table = TRUE,
  show_best_model_details = TRUE,
  show_group_counts = TRUE,
  show_medians = TRUE,
  show_ph_test = TRUE,
  plot.it = FALSE,
  ...
)

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

Arguments

data	Input data frame.
predictor	Continuous predictor variable name (character).
outcome_time	Time-to-event variable name (character).
outcome_event	Event indicator name (0/1) (character).
method	‘"systematic"‘ (max_cuts <= 2) or ‘"genetic"‘.
criterion	‘"AIC"‘, ‘"AICc"‘ or ‘"BIC"‘.
covariates	Character vector of covariate names (optional).
max_cuts	Max number of cut-points to test (non-negative int).
nmin	Min. group size (count or proportion).

seed	Integer or 'NULL'; random seed for 'rgenoud'.
max.generations	Integer; generations for 'rgenoud'. If 'NULL', dynamically scales.
pop.size	Integer; population size for 'rgenoud'. If 'NULL', dynamically scales.
use_cpp	Logical. Automatically checks and calls compiled C++ routines via 'Rcpp'. Default is 'TRUE'.
...	Additional arguments passed down to downstream rendering pipelines.
x	A `find_cutpoint_number_result` object.
object	A `find_cutpoint_number_result` object for analysis overview.
show_comparison_table	Logical. Show information criteria comparison matrix?
show_best_model_details	Logical. Show descriptive layers for the optimal selection?
show_group_counts	Logical. Show categorised patient split breakdowns?
show_medians	Logical. Show Kaplan-Meier time threshold tracking?
show_ph_test	Logical. Display Schoenfeld residuals test?
plot.it	Logical. If `TRUE`, automatically prints the information criterion line chart.
y	Unused mandatory base parameter required for graphic dispatcher pairing inheritance.

Details

'method = "systematic"': grid search respecting 'nmin'. 'method = "genetic"': 'rgenoud' global optimisation. Systematic search is slow for 'max_cuts > 2'; use 'genetic'. Core vector partitions are calculated in compiled C++ via 'Rcpp' for optimal performance.

Value

An S3 object ('find_cutpoint_number_result') with 'results', 'parameters', 'userdata', 'optimal_num_cuts', 'optimal_cuts', and 'candidate_cuts'.

srrstats compliance

.

.

.

.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)

  # Generate a pristine simulated clinical tracking baseline template
```

```
  set.seed(42)
  sim_data <- data.frame(
    time = rexp(40, rate = 0.1),
    event = sample(c(0, 1), 40, replace = TRUE),
    biomarker = rnorm(40, mean = 6, sd = 1.2)
  )

  # Sweep information criteria fit columns up to a 2-cut matrix max
  num_fit <- find_cutpoint_number(
    data = sim_data,
    predictor = "biomarker",
    outcome_time = "time",
    outcome_event = "event",
    max_cuts = 2,
    method = "systematic",
    criterion = "BIC",
    nmin = 5,
    quiet = TRUE
  )
  summary(num_fit)
}
```

`plot.find_cutpoint` *Master S3 Plot Router for find_cutpoint*

Description

Unified plotting dispatch network routing to clinical survival curves, predictor density distributions, hazard ratio forest charts, multi-dimensional objective surfaces, or conditional landmark stratification assets.

Usage

```
## S3 method for class 'find_cutpoint'
plot(
  x,
 type = c("outcome", "distribution", "forest", "surface", "trajectory", "diagnostic",
    "landmark"),
  return_data = FALSE,
  landmark = NULL,
  ...
)
```

Arguments

`x`	A `find_cutpoint` result object.
`type`	Plot framework type: `"outcome"`, `"distribution"`, `"forest"`, `"surface"`, `"trajectory"`, `"diagnostic"`, or `"landmark"`.

return_data	Logical. If TRUE, exits the router early and returns the assigned underlying data frame template.
landmark	Numeric. The operational milestone timestamp used if type = "landmark".
...	Additional arguments passed down to downstream rendering pipelines.

Value

A ggplot canvas object, a multi-panel patchwork collection, or a data.frame if return_data = TRUE.

srrstats compliance

.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  # Build clean local simulation objects with an explicit survival risk split
  set.seed(123)
  mock_df <- data.frame(
    time   = c(runif(15, 50, 100), runif(15, 5, 25)),
    event  = rep(1, 30),
    factor = c(rnorm(15, 5, 0.5), rnorm(15, 15, 0.5))
  )
  res <- find_cutpoint(
    mock_df, "factor", "time", "event",
    num_cuts = 1, method = "systematic", quiet = TRUE, nmin = 3
  )
  p <- plot(res, type = "distribution")
}
```

---

plot_cutpoint_residuals

Diagnostic Plot of Schoenfeld Residuals

Description

High-tier multi-panel diagnostic dashboard tracking the proportional hazards assumption with custom facets per risk cohort stratum.

Usage

```
plot_cutpoint_residuals(x, ...)
```

Arguments

x	A find_cutpoint result object.
...	Unused optional arguments.

Value

A publication-ready ggplot canvas frame, or NULL if the fit fails.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  mock_df <- data.frame(time = 1:30, event = rep(c(0, 1), 15), factor = rnorm(30))
  res <- find_cutpoint(
    mock_df, "factor", "time", "event",
    num_cuts = 1, method = "systematic", quiet = TRUE
  )
  p <- plot_cutpoint_residuals(res)
}
```

---

plot_landmark_stratification

Plot Landmark Stratification Curves

Description

Computes and visualises conditional survival probabilities for patients who survive up to a specified landmark milestone.

Usage

```
plot_landmark_stratification(x, landmark = NULL, ...)
```

Arguments

x	A find_cutpoint result object.
landmark	Numeric value indicating the landmark milestone. If NULL, defaults to 20% of maximum follow-up data timelines.
...	Additional arguments passed down to downstream rendering pipelines.

Value

A ggplot template or survminer survival curve asset mapping layout.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  mock_df <- data.frame(
    time   = runif(25, 5, 50),
    event  = sample(c(0, 1), 25, replace = TRUE),
    factor = rnorm(25, 10, 2)
  )
```

```
  res <- find_cutpoint(
    mock_df, "factor", "time", "event",
    num_cuts = 1, method = "systematic", quiet = TRUE, nmin = 3
  )
  p <- plot_landmark_stratification(res, landmark = 10)
}
```

`plot_optimisation_curve` *Plot Optimisation Curve or Surface from Search*

Description

Plots the metric landscape evaluated across coordinates. Maps a 1D optimisation line for 1-cut systematic setups, or a 2D topographic grid profile for 2-cut layouts.

Usage

```
plot_optimisation_curve(cutpoint_result, ...)
```

Arguments

`cutpoint_result`	A `find_cutpoint` object.
`...`	Unused dots.

Value

A valid `ggplot` object detailing evaluation statistics vs coordinates.

srrstats compliance

.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  # Build clean simulation objects using a 2-cut systematic search
  # to populate the 2D grid matrix log array
  set.seed(123)
  mock_df <- data.frame(
    time   = c(runif(10, 50, 100), runif(10, 20, 60), runif(10, 5, 25)),
    event  = rep(1, 30),
    factor = c(rnorm(10, 2, 0.2), rnorm(10, 7, 0.2), rnorm(10, 15, 0.2))
  )
  res <- find_cutpoint(
    mock_df, "factor", "time", "event",
    num_cuts = 2, method = "systematic", quiet = TRUE, nmin = 3
```

```
  )
  p <- plot_optimisation_curve(res)
}
```

`plot_validation` *Plot Cut-point Optimisation Stability Surface and Help Catalog Page*

Description

Generates a premium, continuous 2D contour surface density topology map tracking the statistical stability of paired discovered cut-points across bootstrap resampling profiles. Automatically handles 1, 2, or multi-cut architectures dynamically.

Usage

```
plot_validation(
  validation_result,
  main = "Resampling Convergence & Stability Landscape",
  focus_cuts = c(1, 2),
  ...
)

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

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

## S3 method for class 'validate_cutpoint_result'
summary(
  object,
  show_descriptives = TRUE,
  show_ci = TRUE,
  show_params = TRUE,
  plot.it = FALSE,
  ...
)
```

Arguments

`validation_result`	A validation object generated by OptSurvCutR containing the validation log history dataset.
`main`	Main title of the chart canvas. Defaults to "Resampling Convergence & Stability Landscape".
`focus_cuts`	A numeric vector of length 2 specifying which two cut-points to map if the model contains more than 2 cuts. Defaults to `c(1, 2)`.

...	Additional arguments passed down to downstream rendering pipelines.
x	A validation result object for generic printing or plotting methods dispatch.
object	A validation result object passed to down-stream summary tracking.
show_descriptives	Logical. Show complete bootstrap distribution descriptives? Defaults to TRUE.
show_ci	Logical. Print 95% Empirical Confidence Interval boundaries? Defaults to TRUE.
show_params	Logical. Display original metadata loop execution parameters? Defaults to TRUE.
plot.it	Logical. If TRUE, automatically outputs the visual sampling line charts. Defaults to FALSE.

Value

A publication-ready ggplot canvas object displaying stability density bounds.

srrstats compliance

.

.

.

Examples

```
mock_val <- list(
  bootstrap_distribution = data.frame(Cut_point_1 = rnorm(30, 10, 1)),
  original_cuts = 10.2,
  parameters = list(predictor = "Biomarker", num_replicates = 30, successful_reps = 30)
)
p <- plot_validation(mock_val)
```

---

theme_optsurv *Custom Clinical Theme for OptSurvCutR*

Description

A clean, publication-ready ggplot2 theme used as the standard aesthetic for all OptSurvCutR plots. Features subtle light-grey background gridlines for high-precision tracing.

Usage

```
theme_optsurv(base_size = 14)
```

Arguments

base_size	Base font size, default is 14.

Value

A ggplot2 theme object containing customised layout parameters.

Examples

```
library(ggplot2)
mock_df <- data.frame(x = 1:10, y = 1:10)
ggplot(mock_df, aes(x, y)) +
  geom_point() +
  theme_optsurv()
```

---

validate_cutpoint *Validate an Optimal Cut-point Using Bootstrapping*

Description

Assesses cut-point stability from find_cutpoint via bootstrap analysis, generating 95% confidence intervals. Streamlined for survival (time-to-event) analysis.

Usage

```
validate_cutpoint(
  cutpoint_result,
  num_replicates = 500,
  n_cores = 1,
  seed = NULL,
  nmin = NULL,
  ...
)
```

Arguments

cutpoint_result	An object from find_cutpoint.
num_replicates	Number of bootstrap replicates. Default is 500.
n_cores	Number of CPU cores to use. Default is 1 (sequential). Set to > 1 to enable parallel processing.
seed	Optional integer for reproducible results.
nmin	Minimum group size for bootstrap runs. Defaults to 90% of original nmin to reduce failures.
...	Additional arguments passed to find_cutpoint (e.g., pop.size, max.generations for genetic algorithm).

Value

An object of class validate_cutpoint_result with original cuts, 95% CIs, bootstrap distribution, and parameters.

srrstats compliance

.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)

  # 1. Create a tiny simulated baseline clinical cohort dataset
  set.seed(123)
  n <- 45
  toy_data <- data.frame(
    time = rexp(n, rate = 0.05),
    event = sample(c(0, 1), n, replace = TRUE, prob = c(0.4, 0.6)),
    marker = rnorm(n, mean = 4, sd = 1.2)
  )

  # 2. Locate initial baseline cut-points via systematic search
  initial_cut <- find_cutpoint(
    data = toy_data, predictor = "marker",
    outcome_time = "time", outcome_event = "event",
    num_cuts = 1, method = "systematic", criterion = "logrank", nmin = 10
  )

  # 3. Run a lightweight bootstrap validation stress-test execution loop
  val_res <- validate_cutpoint(
    cutpoint_result = initial_cut,
    num_replicates = 25, # Small iteration tier for rapid check verification
    n_cores = 1,
    seed = 123
  )

  # 4. Invoke structural S3 output verification hooks
  print(val_res)
  summary(val_res)
  plot(val_res)
}
```

Index

find_cutpoint, 2, *14*
find_cutpoint_number, 5

plot.find_cutpoint, 8
plot.find_cutpoint_number_result
        *(*find_cutpoint_number*)*, 5
plot.validate_cutpoint_result
        *(*plot_validation*)*, 12
plot_cutpoint_residuals, 9
plot_landmark_stratification, 10
plot_optimisation_curve, 11
plot_validation, 12
print.find_cutpoint *(*find_cutpoint*)*, 2
print.find_cutpoint_number_result
        *(*find_cutpoint_number*)*, 5
print.validate_cutpoint_result
        *(*plot_validation*)*, 12

summary.find_cutpoint *(*find_cutpoint*)*, 2
summary.find_cutpoint_number_result
        *(*find_cutpoint_number*)*, 5
summary.validate_cutpoint_result
        *(*plot_validation*)*, 12

theme_optsurv, 13

validate_cutpoint, 14