Package: survHC (via r-universe)

October 22, 2026

Type Package

Title Higher Criticism Tests for Non-Proportional Hazard Deviations

Version 0.1.0

Description Implements omnibus tests for comparing two right-censored survival samples that are powerful against sparse departures from equal hazards, where the hazard difference is concentrated in a small and a priori unknown subset of the follow-up. Exact hypergeometric p-values are computed for each time interval and aggregated with the Higher Criticism statistic of Donoho and Jin (2004) <doi:10.1214/009053604000000265>, or alternatively with the Berk-Jones, Fisher or minimum-p statistics. Global p-values are obtained by permuting group labels, and the intervals driving a detection can be recovered for diagnostic plotting. Tests accept either two samples as vectors or the usual survival formula. The method is described in Kipnis, Galili and Yakhini (2026) <doi:10.1093/biomet/asaf075>, and the companion application note is Kipnis, Galili and Yakhini (2026) <doi:10.48550/arXiv.2610.07268>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

Imports stats, graphics

Suggests survival

URL https://github.com/alonkipnis/lifelines-hc/tree/main/R-package/survHC

BugReports https://github.com/alonkipnis/lifelines-hc/issues

NeedsCompilation no

Author Alon Kipnis [aut, cre], Ben Galili [aut], Zohar Yakhini [aut]

Maintainer Alon Kipnis <alonkipnis@gmail.com>

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

Date/Publication 2026-10-21 17:20:02 UTC

RemoteUrl https://github.com/cran/survHC

RemoteRef HEAD

RemoteSha 0ceba6afe366ffca89f1888c17a64caeb95906f2

Contents

berk_jones_test 2
event_pvalues 5
fisher_combination_test 7
higher_criticism_test 9
min_p_test 12
plot_pvalue_profile 14
suspected_deviations 17

Index **20**

`berk_jones_test` *Berk-Jones test for non-proportional hazard deviations*

Description

Aggregates the same per-interval hypergeometric p-values as `higher_criticism_test` using the exact Berk-Jones statistic, returned as $-\log(BJ)$ so that large values are significant.

Usage

```
berk_jones_test(durations_A, ...)

## Default S3 method:
berk_jones_test(
  durations_A,
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  alternative = c("both", "greater", "less"),
  gamma = 0.2,
  hc_version = c("dj2008", "dj2004", "beta", "star"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  n_permutations = 0,
  seed = NULL,
  ...
)

## S3 method for class 'formula'
berk_jones_test(formula, data, subset, na.action, ...)
```

Arguments

`durations_A`, `durations_B`
: Numeric vectors of event or censoring times. `durations_A` may instead be a `formula`, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`alternative`
: One of `"both"` (default), `"greater"` (excess hazard in group B) or `"less"` (excess hazard in group A).

`gamma`
: Fraction of the smallest ordered p-values scanned, in (0, 1]. Default 0.2.

`hc_version`
: Which Higher Criticism variant to use, matching the named methods of the Python package `multitest` (>= 0.2.2) and the `hc_version` argument of `lifelines-hc`. `"dj2008"` (default) uses $u_i = i/N$ with denominator $\sqrt{u_i(1-u_i)/N}$; `"dj2004"` uses $u_i = i/N$ with denominator $\sqrt{p_{(i)}(1-p_{(i)})/N}$, which is equation (2) of Kipnis, Galili and Yakhini (2026); `"beta"` uses $u_i = i/(N+1)$ with denominator $\sqrt{u_i(1-u_i)/(N+2)}$; `"star"` uses the `"beta"` denominator but starts the scan at the first ordered p-value exceeding $(1-\epsilon)/N$. Ignored by `berk_jones_test`, `fisher_combination_test` and `min_p_test`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. `NULL` (default) applies no pooling.

`n_permutations`
: Number of label permutations used to calibrate the p-value. `0` (default) skips calibration and returns `NA`.

`seed`
: Optional integer seed for the permutation test. When supplied, the caller's random seed is restored on exit.

`formula`
: A two-sided formula `Surv(time, status) ~ group`; see Details.

`data`
: An optional data frame, list or environment holding the variables named in `formula`.

`subset`
: An optional vector specifying a subset of observations to use.

`na.action`
: A function describing what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`, normally `na.omit`.

`...`
: Further arguments. The formula method passes these on to the default method, which accepts none and signals an error if any are supplied, so that a misspelled argument name cannot be silently ignored.

Details

The statistic is `Inf` when the exact Berk-Jones p-value underflows to zero, which happens routinely on unpooled continuous survival data where the smallest per-interval p-value is extreme. Set

`n_intervals_to_pool` to obtain a finite value. This matches the behaviour of the reference Python implementation.

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package `lifelines-hc`. The formula method takes the usual survival formula `Surv(time, status) ~ group`, which needs the **survival** package for `Surv` itself; without it, `cbind(time, status) ~ group` and `time ~ group` also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so `alternative = "greater"` tests for excess hazard in the second level. Levels are taken in factor order for a factor, and otherwise in the order `sort(unique(.))` gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects `NA` durations; the formula method applies `na.action`, which by default drops incomplete rows.

Value

An object of class `"htest"`.

References

Moscovich, A., Nadler, B. and Spiegelman, C. (2016). On the exact Berk-Jones statistics and their p-value calculation. *Electronic Journal of Statistics* **10**(2), 2329–2354.

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Kipnis, A., Galili, B. and Yakhini, Z. (2026). `lifelines-hc`: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

`higher_criticism_test`

Examples

```
set.seed(1)
berk_jones_test(rexp(200, 1 / 10), rexp(200, 1 / 10), n_intervals_to_pool = 50)

trial <- data.frame(time = rexp(400, 1 / 10), status = 1,
                    arm = rep(c("A", "B"), each = 200))
berk_jones_test(cbind(time, status) ~ arm, data = trial,
                n_intervals_to_pool = 50)
```

event_pvalues *Per-interval hypergeometric p-values*

Description

Returns the exact one-sided hypergeometric p-value for each event time, or for each pooled interval when `n_intervals_to_pool` is supplied. Each p-value quantifies the evidence against equal hazards at that point in the follow-up.

Usage

```
event_pvalues(durations_A, ...)

## Default S3 method:
event_pvalues(
  durations_A,
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  alternative = c("both", "greater", "less"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  ...
)

## S3 method for class 'formula'
event_pvalues(formula, data, subset, na.action, ...)
```

Arguments

`durations_A`, `durations_B`
: Numeric vectors of event or censoring times. `durations_A` may instead be a `formula`, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`alternative`
: One of `"both"` (default), returning a list with components `greater` and `less`; `"greater"`; or `"less"`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. `NULL` (default) applies no pooling.

`formula`	A two-sided formula `Surv(time, status) ~ group`; see Details.
`data`	An optional data frame, list or environment holding the variables named in `formula`.
`subset`	An optional vector specifying a subset of observations to use.
`na.action`	A function describing what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`, normally `na.omit`.
`...`	Further arguments. The formula method passes these on to the default method, which accepts none and signals an error if any are supplied, so that a misspelled argument name cannot be silently ignored.

Details

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package `lifelines-hc`. The formula method takes the usual survival formula `Surv(time, status) ~ group`, which needs the **survival** package for `Surv` itself; without it, `cbind(time, status) ~ group` and `time ~ group` also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so `alternative = "greater"` tests for excess hazard in the second level. Levels are taken in factor order for a factor, and otherwise in the order `sort(unique(.))` gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects `NA` durations; the formula method applies `na.action`, which by default drops incomplete rows.

Value

A numeric vector, or a list of two numeric vectors when `alternative = "both"`.

References

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Kipnis, A., Galili, B. and Yakhini, Z. (2026). `lifelines-hc`: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

`suspected_deviations`

Examples

```
set.seed(1)
p <- event_pvalues(rexp(200, 1 / 10), rexp(200, 1 / 10),
                   alternative = "greater", n_intervals_to_pool = 50)
summary(p)

trial <- data.frame(time = rexp(400, 1 / 10), status = 1,
                    arm = rep(c("A", "B"), each = 200))
```

```
summary(event_pvalues(cbind(time, status) ~ arm, data = trial,
                      alternative = "greater", n_intervals_to_pool = 50))
```

---

fisher_combination_test

Fisher combination test for non-proportional hazard deviations

---

Description

Aggregates the same per-interval hypergeometric p-values as higher_criticism_test using Fisher's method, $-2\sum \log p_i$.

Usage

```
fisher_combination_test(durations_A, ...)

## Default S3 method:
fisher_combination_test(
  durations_A,
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  alternative = c("both", "greater", "less"),
  gamma = 0.2,
  hc_version = c("dj2008", "dj2004", "beta", "star"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  n_permutations = 0,
  seed = NULL,
  ...
)

## S3 method for class 'formula'
fisher_combination_test(
  formula,
  data,
  subset,
  na.action,
  ...
)
```

Arguments

durations_A, durations_B

Numeric vectors of event or censoring times. durations_A may instead be a formula, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`alternative`
: One of `"both"` (default), `"greater"` (excess hazard in group B) or `"less"` (excess hazard in group A).

`gamma`
: Fraction of the smallest ordered p-values scanned, in (0, 1]. Default 0.2.

`hc_version`
: Which Higher Criticism variant to use, matching the named methods of the Python package `multitest` (>= 0.2.2) and the `hc_version` argument of `lifelines-hc`. `"dj2008"` (default) uses $u_i = i/N$ with denominator $\sqrt{u_i(1-u_i)/N}$; `"dj2004"` uses $u_i = i/N$ with denominator $\sqrt{p_{(i)}(1-p_{(i)})/N}$, which is equation (2) of Kipnis, Galili and Yakhini (2026); `"beta"` uses $u_i = i/(N+1)$ with denominator $\sqrt{u_i(1-u_i)/(N+2)}$; `"star"` uses the `"beta"` denominator but starts the scan at the first ordered p-value exceeding $(1-\epsilon)/N$. Ignored by `berk_jones_test`, `fisher_combination_test` and `min_p_test`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. `NULL` (default) applies no pooling.

`n_permutations`
: Number of label permutations used to calibrate the p-value. `0` (default) skips calibration and returns `NA`.

`seed`
: Optional integer seed for the permutation test. When supplied, the caller's random seed is restored on exit.

`formula`
: A two-sided formula `Surv(time, status) ~ group`; see Details.

`data`
: An optional data frame, list or environment holding the variables named in `formula`.

`subset`
: An optional vector specifying a subset of observations to use.

`na.action`
: A function describing what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`, normally `na.omit`.

`...`
: Further arguments. The formula method passes these on to the default method, which accepts none and signals an error if any are supplied, so that a misspelled argument name cannot be silently ignored.

Details

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package `lifelines-hc`. The formula method takes the usual survival formula `Surv(time, status) ~ group`, which needs the **survival** package for `Surv` itself; without it, `cbind(time, status) ~ group` and `time ~ group` also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so `alternative = "greater"` tests for excess hazard in the second level. Levels are taken

in factor order for a factor, and otherwise in the order `sort(unique(.))` gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects `NA` durations; the formula method applies `na.action`, which by default drops incomplete rows.

Value

An object of class `"htest"`.

References

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Kipnis, A., Galili, B. and Yakhini, Z. (2026). `lifelines-hc`: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

`higher_criticism_test`

Examples

```
set.seed(1)
fisher_combination_test(rexp(200, 1 / 10), rexp(200, 1 / 10),
                        n_intervals_to_pool = 50)

trial <- data.frame(time = rexp(400, 1 / 10), status = 1,
                    arm = rep(c("A", "B"), each = 200))
fisher_combination_test(cbind(time, status) ~ arm, data = trial,
                        n_intervals_to_pool = 50)
```

`higher_criticism_test` *Higher Criticism test for non-proportional hazard deviations*

Description

Compares two right-censored survival samples using Higher Criticism applied to exact per-interval hypergeometric p-values. The test is powerful against *sparse* departures, where the hazard difference is concentrated in a small and a priori unknown subset of the follow-up, a regime in which the log-rank test has little power.

Usage

```
higher_criticism_test(durations_A, ...)

## Default S3 method:
higher_criticism_test(
  durations_A,
```

```
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  alternative = c("both", "greater", "less"),
  gamma = 0.2,
  hc_version = c("dj2008", "dj2004", "beta", "star"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  n_permutations = 0,
  seed = NULL,
  ...
)

## S3 method for class 'formula'
higher_criticism_test(formula, data, subset, na.action, ...)
```

Arguments

`durations_A`, `durations_B`
: Numeric vectors of event or censoring times. `durations_A` may instead be a `formula`, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`alternative`
: One of `"both"` (default), `"greater"` (excess hazard in group B) or `"less"` (excess hazard in group A).

`gamma`
: Fraction of the smallest ordered p-values scanned, in (0, 1]. Default 0.2.

`hc_version`
: Which Higher Criticism variant to use, matching the named methods of the Python package `multitest` (>= 0.2.2) and the `hc_version` argument of `lifelines-hc`. `"dj2008"` (default) uses $u_i = i/N$ with denominator $\sqrt{u_i(1-u_i)/N}$; `"dj2004"` uses $u_i = i/N$ with denominator $\sqrt{p_{(i)}(1-p_{(i)})/N}$, which is equation (2) of Kipnis, Galili and Yakhini (2026); `"beta"` uses $u_i = i/(N+1)$ with denominator $\sqrt{u_i(1-u_i)/(N+2)}$; `"star"` uses the `"beta"` denominator but starts the scan at the first ordered p-value exceeding $(1-\epsilon)/N$. Ignored by `berk_jones_test`, `fisher_combination_test` and `min_p_test`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. `NULL` (default) applies no pooling.

`n_permutations`
: Number of label permutations used to calibrate the p-value. `0` (default) skips calibration and returns `NA`.

`seed`
: Optional integer seed for the permutation test. When supplied, the caller's random seed is restored on exit.

`formula`	A two-sided formula `Surv(time, status) ~ group`; see Details.
`data`	An optional data frame, list or environment holding the variables named in `formula`.
`subset`	An optional vector specifying a subset of observations to use.
`na.action`	A function describing what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`, normally `na.omit`.
`...`	Further arguments. The formula method passes these on to the default method, which accepts none and signals an error if any are supplied, so that a misspelled argument name cannot be silently ignored.

Details

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package `lifelines-hc`. The formula method takes the usual survival formula `Surv(time, status) ~ group`, which needs the **survival** package for `Surv` itself; without it, `cbind(time, status) ~ group` and `time ~ group` also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so `alternative = "greater"` tests for excess hazard in the second level. Levels are taken in factor order for a factor, and otherwise in the order `sort(unique(.))` gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects `NA` durations; the formula method applies `na.action`, which by default drops incomplete rows.

Value

An object of class `"htest"`, with the Higher Criticism score in `statistic` and the permutation p-value in `p.value`. The permutation statistics are returned in `permutation_statistics`. The formula method additionally records the two group levels, in the order (A, B), in `group_levels`.

References

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Donoho, D. and Jin, J. (2004). Higher criticism for detecting sparse heterogeneous mixtures. *The Annals of Statistics* **32**(3), 962–994. doi:10.1214/009053604000000265

Kipnis, A., Galili, B. and Yakhini, Z. (2026). `lifelines-hc`: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

`suspected_deviations` to recover the intervals driving a detection; `berk_jones_test`, `fisher_combination_test` and `min_p_test` for alternative aggregations of the same per-interval p-values.

Examples

```
set.seed(1)
T_A <- rexp(200, rate = 1 / 10)
T_B <- rexp(200, rate = 1 / 10)
higher_criticism_test(T_A, T_B, n_intervals_to_pool = 50)

# with permutation calibration
higher_criticism_test(T_A, T_B, n_intervals_to_pool = 50,
                      n_permutations = 200, seed = 42)

# the same data through the formula interface
trial <- data.frame(time = c(T_A, T_B), status = 1,
                    arm = rep(c("A", "B"), c(200, 200)))
higher_criticism_test(cbind(time, status) ~ arm, data = trial,
                      n_intervals_to_pool = 50)

# Surv() needs the survival package, which is only suggested
if (requireNamespace("survival", quietly = TRUE)) {
  higher_criticism_test(survival::Surv(time, status) ~ arm, data = trial,
                        n_intervals_to_pool = 50)
}
```

`min_p_test` *Minimum-p test for non-proportional hazard deviations*

Description

Aggregates the same per-interval hypergeometric p-values as `higher_criticism_test` by taking the minimum, reported as $-\log(\min_i p_i)$.

Usage

```
min_p_test(durations_A, ...)

## Default S3 method:
min_p_test(
  durations_A,
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  alternative = c("both", "greater", "less"),
  gamma = 0.2,
  hc_version = c("dj2008", "dj2004", "beta", "star"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  n_permutations = 0,
  seed = NULL,
  ...
```

```
)

## S3 method for class 'formula'
min_p_test(formula, data, subset, na.action, ...)
```

Arguments

`durations_A`, `durations_B`
: Numeric vectors of event or censoring times. `durations_A` may instead be a `formula`, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`alternative`
: One of `"both"` (default), `"greater"` (excess hazard in group B) or `"less"` (excess hazard in group A).

`gamma`
: Fraction of the smallest ordered p-values scanned, in (0, 1]. Default 0.2.

`hc_version`
: Which Higher Criticism variant to use, matching the named methods of the Python package `multitest` (>= 0.2.2) and the `hc_version` argument of `lifelines-hc`. `"dj2008"` (default) uses $u_i = i/N$ with denominator $\sqrt{u_i(1-u_i)/N}$; `"dj2004"` uses $u_i = i/N$ with denominator $\sqrt{p_{(i)}(1-p_{(i)})/N}$, which is equation (2) of Kipnis, Galili and Yakhini (2026); `"beta"` uses $u_i = i/(N+1)$ with denominator $\sqrt{u_i(1-u_i)/(N+2)}$; `"star"` uses the `"beta"` denominator but starts the scan at the first ordered p-value exceeding $(1-\epsilon)/N$. Ignored by `berk_jones_test`, `fisher_combination_test` and `min_p_test`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. `NULL` (default) applies no pooling.

`n_permutations`
: Number of label permutations used to calibrate the p-value. `0` (default) skips calibration and returns `NA`.

`seed`
: Optional integer seed for the permutation test. When supplied, the caller's random seed is restored on exit.

`formula`
: A two-sided formula `Surv(time, status) ~ group`; see Details.

`data`
: An optional data frame, list or environment holding the variables named in `formula`.

`subset`
: An optional vector specifying a subset of observations to use.

`na.action`
: A function describing what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`, normally `na.omit`.

`...`
: Further arguments. The formula method passes these on to the default method, which accepts none and signals an error if any are supplied, so that a misspelled argument name cannot be silently ignored.

Details

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package `lifelines-hc`. The formula method takes the usual survival formula `Surv(time, status) ~ group`, which needs the **survival** package for `Surv` itself; without it, `cbind(time, status) ~ group` and `time ~ group` also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so `alternative = "greater"` tests for excess hazard in the second level. Levels are taken in factor order for a factor, and otherwise in the order `sort(unique(.))` gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects `NA` durations; the formula method applies `na.action`, which by default drops incomplete rows.

Value

An object of class `"htest"`.

References

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Kipnis, A., Galili, B. and Yakhini, Z. (2026). `lifelines-hc`: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

`higher_criticism_test`

Examples

```
set.seed(1)
min_p_test(rexp(200, 1 / 10), rexp(200, 1 / 10), n_intervals_to_pool = 50)

trial <- data.frame(time = rexp(400, 1 / 10), status = 1,
                    arm = rep(c("A", "B"), each = 200))
min_p_test(cbind(time, status) ~ arm, data = trial, n_intervals_to_pool = 50)
```

`plot_pvalue_profile` *Plot the per-interval p-value profile*

Description

Draws the signed $-\log_{10}$ per-interval hypergeometric p-value profile, with bars above the axis marking excess events in group B and bars below marking excess in group A. Intervals flagged by the Higher Criticism threshold are highlighted, giving a visual reading of which windows of the follow-up drive a detection.

Usage

```
plot_pvalue_profile(durations_A, ...)

## Default S3 method:
plot_pvalue_profile(
  durations_A,
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  gamma = 0.2,
  hc_version = c("dj2008", "dj2004", "beta", "star"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  col_flagged = "firebrick",
  col_plain = "steelblue",
  xlab = "Time",
  ylab = "signed -log10(p)",
  main = NULL,
  ...
)

## S3 method for class 'formula'
plot_pvalue_profile(formula, data, subset, na.action, ...)
```

Arguments

`durations_A`, `durations_B`
: Numeric vectors of event or censoring times. `durations_A` may instead be a `formula`, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`gamma`
: Fraction of the smallest ordered p-values scanned, in (0, 1]. Default 0.2.

`hc_version`
: Which Higher Criticism variant to use, matching the named methods of the Python package `multitest` (>= 0.2.2) and the `hc_version` argument of `lifelines-hc`. `"dj2008"` (default) uses $u_i = i/N$ with denominator $\sqrt{u_i(1-u_i)/N}$; `"dj2004"` uses $u_i = i/N$ with denominator $\sqrt{p_{(i)}(1-p_{(i)})/N}$, which is equation (2) of Kipnis, Galili and Yakhini (2026); `"beta"` uses $u_i = i/(N+1)$ with denominator $\sqrt{u_i(1-u_i)/(N+2)}$; `"star"` uses the `"beta"` denominator but starts the scan at the first ordered p-value exceeding $(1-\epsilon)/N$. Ignored by `berk_jones_test`, `fisher_combination_test` and `min_p_test`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most

one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. NULL (default) applies no pooling.

col_flagged — Colour for intervals at or below the HC threshold.

col_plain — Colour for the remaining intervals.

xlab, ylab, main — Standard graphical parameters.

formula — A two-sided formula Surv(time, status) ~ group; see Details.

data — An optional data frame, list or environment holding the variables named in formula.

subset — An optional vector specifying a subset of observations to use.

na.action — A function describing what should happen when the data contain NAs. Defaults to getOption("na.action"), normally na.omit.

... — Further arguments passed to plot.default.

Details

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package lifelines-hc. The formula method takes the usual survival formula Surv(time, status) ~ group, which needs the **survival** package for Surv itself; without it, cbind(time, status) ~ group and time ~ group also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so alternative = "greater" tests for excess hazard in the second level. Levels are taken in factor order for a factor, and otherwise in the order sort(unique(.)) gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects NA durations; the formula method applies na.action, which by default drops incomplete rows.

Value

Invisibly, the data frame returned by suspected_deviations with alternative = "both".

References

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Kipnis, A., Galili, B. and Yakhini, Z. (2026). lifelines-hc: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

suspected_deviations

Examples

```
set.seed(1)
T_A <- rexp(300, 1 / 10)
T_B <- c(rexp(250, 1 / 10), rexp(50, 1 / 2))
plot_pvalue_profile(T_A, T_B, n_intervals_to_pool = 50)

# the same plot through the formula interface
trial <- data.frame(time = c(T_A, T_B), status = 1,
                    arm = rep(c("A", "B"), c(300, 300)))
plot_pvalue_profile(cbind(time, status) ~ arm, data = trial,
                    n_intervals_to_pool = 50)
```

---

suspected_deviations *Intervals with suspected non-proportional hazard deviations*

Description

Computes per-interval hypergeometric p-values and flags those at or below the Higher Criticism threshold. This localises a detection to specific windows of the follow-up, which is the diagnostic information a single-number test cannot provide, and is suitable for shading on a Kaplan-Meier plot.

Usage

```
suspected_deviations(durations_A, ...)

## Default S3 method:
suspected_deviations(
  durations_A,
  durations_B,
  event_observed_A = NULL,
  event_observed_B = NULL,
  alternative = c("greater", "less", "both"),
  gamma = 0.2,
  hc_version = c("dj2008", "dj2004", "beta", "star"),
  t_0 = -1,
  n_intervals_to_pool = NULL,
  ...
)

## S3 method for class 'formula'
suspected_deviations(formula, data, subset, na.action, ...)
```

Arguments

`durations_A`, `durations_B`
: Numeric vectors of event or censoring times. `durations_A` may instead be a `formula`, which selects the formula method.

`event_observed_A`, `event_observed_B`
: Optional 0/1 vectors, 1 indicating an observed event and 0 a censored observation. Default: all events.

`alternative`
: One of `"greater"` (default), `"less"` or `"both"`. With `"both"`, columns for the reverse direction are added and `suspected` flags either direction.

`gamma`
: Fraction of the smallest ordered p-values scanned, in (0, 1]. Default 0.2.

`hc_version`
: Which Higher Criticism variant to use, matching the named methods of the Python package `multitest` (>= 0.2.2) and the `hc_version` argument of `lifelines-hc`. `"dj2008"` (default) uses $u_i = i/N$ with denominator $\sqrt{u_i(1-u_i)/N}$; `"dj2004"` uses $u_i = i/N$ with denominator $\sqrt{p_{(i)}(1-p_{(i)})/N}$, which is equation (2) of Kipnis, Galili and Yakhini (2026); `"beta"` uses $u_i = i/(N+1)$ with denominator $\sqrt{u_i(1-u_i)/(N+2)}$; `"star"` uses the `"beta"` denominator but starts the scan at the first ordered p-value exceeding $(1-\epsilon)/N$. Ignored by `berk_jones_test`, `fisher_combination_test` and `min_p_test`.

`t_0`
: Restrict the analysis to events at or before `t_0`. A negative value (default) applies no restriction.

`n_intervals_to_pool`
: Number of equal-width intervals to pool event times into. Strongly recommended for continuous survival data, where each distinct time carries at most one event and the per-time hypergeometric p-values are too coarse to be informative. Values between 50 and 200 usually work well. `NULL` (default) applies no pooling.

`formula`
: A two-sided formula `Surv(time, status) ~ group`; see Details.

`data`
: An optional data frame, list or environment holding the variables named in `formula`.

`subset`
: An optional vector specifying a subset of observations to use.

`na.action`
: A function describing what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`, normally `na.omit`.

`...`
: Further arguments. The formula method passes these on to the default method, which accepts none and signals an error if any are supplied, so that a misspelled argument name cannot be silently ignored.

Details

Two interfaces are available and return the same numbers. The default method takes the two samples as vectors, mirroring the Python package `lifelines-hc`. The formula method takes the usual survival formula `Surv(time, status) ~ group`, which needs the **survival** package for `Surv` itself; without it, `cbind(time, status) ~ group` and `time ~ group` also work, the latter treating every observation as an event.

The grouping variable must take exactly two values. Its first level is group A and its second is group B, so `alternative = "greater"` tests for excess hazard in the second level. Levels are taken in factor order for a factor, and otherwise in the order `sort(unique(.))` gives: numerically for numbers, alphabetically for strings.

The two interfaces differ only in their treatment of missing values. The default method rejects `NA` durations; the formula method applies `na.action`, which by default drops incomplete rows.

Value

A data frame with one row per event time or pooled interval and columns `time` (or `time_lower` and `time_upper` when pooled), `at_risk_A`, `at_risk_B`, `observed_A`, `observed_B`, `hypergeom_pvalue`, `suspected` and `hc_threshold`.

References

Kipnis, A., Galili, B. and Yakhini, Z. (2026). Higher criticism for rare and weak non-proportional hazard deviations in survival analysis. *Biometrika* **113**(1), asaf075. doi:10.1093/biomet/asaf075

Kipnis, A., Galili, B. and Yakhini, Z. (2026). `lifelines-hc`: Higher Criticism testing for sparse non-proportional hazard departures in Python. doi:10.48550/arXiv.2610.07268

See Also

`higher_criticism_test`, `plot_pvalue_profile`

Examples

```
set.seed(1)
# a sparse departure confined to one window
T_A <- rexp(300, 1 / 10)
T_B <- c(rexp(250, 1 / 10), rexp(50, 1 / 2))
d <- suspected_deviations(T_A, T_B, n_intervals_to_pool = 50)
head(d[d$suspected, ])

trial <- data.frame(time = c(T_A, T_B), status = 1,
                    arm = rep(c("A", "B"), c(300, 300)))
d2 <- suspected_deviations(cbind(time, status) ~ arm, data = trial,
                           n_intervals_to_pool = 50)
head(d2[d2$suspected, ])
```

Index

berk_jones_test, 2, *11*

event_pvalues, 5

fisher_combination_test, 7, *11*
formula, *3*, *5*, *7*, *10*, *13*, *15*, *17*

higher_criticism_test, *2*, *4*, *7*, 9, *9*, *12*, *14*, *19*

min_p_test, *11*, 12

na.omit, *3*, *6*, *8*, *11*, *13*, *16*, *18*

plot.default, *16*
plot_pvalue_profile, 14, *19*

suspected_deviations, *6*, *11*, *16*, 17