Package: clusteredMSM (via r-universe)

October 24, 2026

Type Package

Title Nonparametric Analysis of Clustered Multistate Processes

Version 0.1.0

Description Nonparametric estimation of population-averaged transition probabilities, with cluster-bootstrap pointwise confidence intervals, simultaneous confidence bands, and two-sample Kolmogorov-Smirnov-type tests for clustered or independent multistate process data. Estimation follows Bakoyannis (2021) <doi:10.1111/biom.13327>; two-sample inference for the cluster-randomized and independent-samples designs follows Bakoyannis and Bandyopadhyay (2022) <doi:10.1007/s10463-021-00819-x>. Both methods use the working-independence Aalen-Johansen estimator. The package supports both progressive (acyclic) and non-monotone (e.g., illness-death with recovery) multistate processes, right censoring, left truncation, and informative cluster size. The user supplies data in interval format (one row per mutually-exclusive time interval per subject) and interacts with the package through a single formula-based function, patp().

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports survival, stats, utils

Suggests mstate, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/gbakoyannis/clusteredMSM

BugReports https://github.com/gbakoyannis/clusteredMSM/issues

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3
Repository https://gbakoyannis.r-universe.dev
Date/Publication 2026-05-20 02:02:38 UTC
RemoteUrl https://github.com/gbakoyannis/clusteredmsm
RemoteRef HEAD
RemoteSha 96de6d1225090939789054df72da4190b1cb713c

Contents

ci_cloglog . . . 2
cluster_boot . . . 3
confidence_band . . . 5
cut_at_lm . . . 6
example_msm . . . 7
fit_chaz . . . 8
intervals_to_long . . . 9
ks_pvalue . . . 10
msm . . . 11
patp . . . 12
print.patp . . . 15
prodint_AJ . . . 16
state_at . . . 17
summary.patp . . . 18
trans_mat . . . 18

Index **20**

`ci_cloglog` *Pointwise Confidence Intervals via Complementary Log-Log Transformation*

Description

Computes pointwise confidence intervals for a transition probability estimate using the cloglog transformation $g(p) = \log(-\log p)$ with a delta-method standard error scaling. The bootstrap is run on the original probability scale; the cloglog SE is then derived analytically.

Usage

```
ci_cloglog(point, se, level = 0.95)
```

Arguments

`point`	Numeric vector of point estimates in (0, 1), length T.
`se`	Numeric vector of bootstrap standard errors of `point` on the probability scale, same length as `point` (i.e. `apply(boot_matrix, 1, sd, na.rm = TRUE)`).
`level`	Confidence level. Default 0.95.

Details

By the delta method, $\mathrm{SE}(g(\hat{P})) = \mathrm{SE}(\hat{P})/|\hat{P}\log\hat{P}|$ (here $|g'(p)| = 1/|p\log p|$ on (0,1)). The interval is built symmetrically on the cloglog scale and back-transformed via $p = \exp(-\exp(\cdot))$. Because g is monotone decreasing on (0, 1), the upper end on the cloglog scale becomes the *lower* end on the probability scale, and vice versa.

For point estimates exactly equal to 0 or 1 the cloglog is undefined and `NA` is returned at those positions.

Value

A list with elements `ll` and `ul`: numeric vectors of lower and upper confidence limits, the same length as `point`.

`cluster_boot` *Generic Cluster Bootstrap Engine*

Description

Performs the nonparametric cluster bootstrap: resamples whole clusters with replacement and applies a user-supplied statistic function to each bootstrap replicate.

Usage

```
cluster_boot(
  data,
  cluster,
  B,
  fn,
  ...,
  strata = NULL,
  seed = NULL,
  verbose = FALSE
)
```

Arguments

`data`	A data frame containing the cluster ID column named by `cluster`. All other columns are passed through unchanged to `fn`.
`cluster`	Character. Name of the cluster ID column.
`B`	Integer. Number of bootstrap replications.
`fn`	Function. Called as `fn(boot_data, ...)` once per bootstrap replicate. Must return a numeric vector of fixed length (the length must be the same across replicates – typically achieved by evaluating the statistic on a fixed grid of times).
`...`	Additional arguments passed to `fn`.

strata	Optional named vector mapping each unique cluster ID (as character) to a stratum label. When supplied, resampling is stratified: in each replicate, the original number of clusters in each stratum is drawn with replacement from the clusters in that stratum (so per-stratum cluster counts are preserved across replicates). When NULL (default), unstratified resampling is used.
seed	Optional integer. If non-NULL, set.seed(seed) is called before bootstrapping for reproducibility.
verbose	Logical. If TRUE, prints a progress message every 100 replicates. Default FALSE.

Details

The bootstrap is nonparametric and at the cluster level: in each replicate, n clusters are sampled with replacement (where n is the number of unique values of data[[cluster]]), and the resulting cluster-bound rows are stacked into a new data frame.

Resampled clusters are re-IDed (1, 2, ..., n) so that downstream code treating cluster IDs as distinct labels (e.g., tapply aggregations) works correctly even when the same original cluster appears multiple times in a replicate.

Stratified resampling (strata) supports cluster-randomized designs in which each cluster belongs to a single group (case ii of Bakoyannis & Bandyopadhyay 2022): the per-stratum cluster counts are fixed across replicates, so a replicate can never wipe out one of the groups.

The bootstrap is statistic-agnostic: the same engine is used for point-estimator standard errors, two-sample tests, and confidence bands. Whatever fn returns is what gets bootstrapped.

Value

A numeric matrix with length(fn(data, ...)) rows and B columns. Column b is the result of fn applied to bootstrap replicate b.

Examples

```
data(example_msm)

# Cluster-bootstrap a simple summary: the mean of Tstop across
# rows. The user-supplied fn can return any fixed-length numeric
# vector; the typical use case is a transition probability curve
# evaluated on a fixed time grid (which is exactly what patp() does
# under the hood).
boot <- cluster_boot(
  data = example_msm, cluster = "cluster", B = 50,
  fn   = function(d) mean(d$Tstop),
  seed = 1
)
c(point = mean(boot, na.rm = TRUE),
  se    = stats::sd(boot, na.rm = TRUE))
```

`confidence_band` *Simultaneous Confidence Band via Cluster-Bootstrap Quantile*

Description

Constructs a simultaneous (uniform-over-time) confidence band for a transition probability curve. The band is built on the cloglog scale: for each bootstrap replicate, the standardized supremum $\sup_t |g(\hat{P}_b(t)) - g(\hat{P}(t))|/\mathrm{SE}_g(t)$ is computed, where $\mathrm{SE}_g(t)$ is the delta-method SE on the cloglog scale. The `level` quantile of those suprema is the critical value, and the band is back-transformed to the probability scale.

Usage

```
confidence_band(point, boot, times, level = 0.95, trim = c(0.05, 0.95))
```

Arguments

`point`	Numeric vector of point estimates over a time grid.
`boot`	Numeric matrix of bootstrap replicates of `point` on the original probability scale: rows correspond to time points (matching `point`), columns to bootstrap replicates.
`times`	Numeric vector of times matching the rows of `boot` and the entries of `point`.
`level`	Confidence level. Default 0.95.
`trim`	Numeric vector of length 2. Lower and upper quantiles of the jump-time distribution at which to clip the band (avoids the "fans out" behavior near the extremes of follow-up). Default `c(0.05, 0.95)`.

Details

Only one bootstrap on the original (probability) scale is required: $\mathrm{SE}_g(t)$ is obtained by the delta method from $\mathrm{SE}(\hat{P}(t)) = \mathrm{sd}(\hat{P}^*(t))$, and the cloglog values $g(\hat{P}_b(t))$ are computed by transforming the existing replicates pointwise – no separate cloglog-scale bootstrap is run.

Value

A list with elements `ll.band` and `ul.band`: numeric vectors of band limits, with `NA` outside the trimmed range or where `point` is at the boundary 0/1.

Note

This version uses an equal-precision-type band on the cloglog scale, in which the bootstrap deviation is standardized pointwise by the delta-method standard error of $P(t)$. By the self- cancellation of $\sqrt{n}$ factors, this is asymptotically equivalent to standardizing the limit process by its asymptotic standard error on the cloglog scale, which is the standard equal-precision construction. This construction is asymptotically valid under conditions C1-C6 of Bakoyannis (2021), but differs from the Hall-Wellner-type weight proposed in Section 2.3 of that paper. Both constructions yield asymptotically valid simultaneous bands. The paper's Hall-Wellner-type construction will be added as a `band_method` option in a future release and will become the default.

References

Bakoyannis, G. (2021). Nonparametric analysis of nonhomogeneous multistate processes with clustered observations. *Biometrics*, 77(2), 533-546. doi:10.1111/biom.13327

`cut_at_lm` *Truncate a Long-Format Multistate Dataset at a Landmark Time*

Description

Restricts a long-format multistate dataset to follow-up after a landmark time `s`. Intervals ending before `s` are removed; intervals straddling `s` are truncated to start at `s`.

Usage

```
cut_at_lm(data, s)
```

Arguments

`data`	A long-format data frame, typically produced by `intervals_to_long()`, with columns `Tstart`, `Tstop`, `from`, `to`, `trans`, `status`, `id`, and (optionally) `cluster`.
`s`	Numeric scalar. The landmark time.

Details

This function is used in the landmark Aalen-Johansen estimator, which relaxes the Markov assumption by re-fitting hazards on the cohort still under observation at time `s`.

Important: this function only truncates the time axis. To compute the landmark estimator, you must also restrict to subjects who are in the relevant starting state at time `s` – use `state_at()` for that step.

Intervals where `Tstop == s` exactly are dropped (the subject has already transitioned by `s`); intervals where `Tstart >= s` are kept unchanged; intervals straddling `s` (`Tstart < s < Tstop`) have their `Tstart` reset to `s`. If a straddling interval was going to end in an event (`status == 1`) at `Tstop > s`, the event is preserved.

Value

A data frame of the same shape as `data`, restricted and truncated as described above.

References

Putter H, Spitoni C (2018). Non-parametric estimation of transition probabilities in non-Markov multi-state models: the landmark Aalen-Johansen estimator. *Statistical Methods in Medical Research* 27(7):2081-2092. doi:10.1177/0962280216674497

Bakoyannis G (2021). Nonparametric analysis of nonhomogeneous multistate processes with clustered observations. *Biometrics* 77(2):533-546. doi:10.1111/biom.13327

Examples

```
data(example_msm)
tmat <- trans_mat(list(c(2, 3), c(1, 3), integer(0)),
                  names = c("Healthy", "Ill", "Dead"))
long <- intervals_to_long(example_msm, tmat)

# Restrict follow-up to after the landmark time s = 1.5
long_lm <- cut_at_lm(long, s = 1.5)
head(long_lm)
```

`example_msm` *Synthetic Clustered Illness-Death-with-Recovery Data*

Description

A synthetic interval-format multistate dataset used in package examples, the README, and the vignette. The data are generated from an illness-death-with-recovery process with cluster-level frailty, so they exercise the package's headline feature: support for non-monotone (cyclic) transitions.

Usage

```
example_msm
```

Format

A data frame with 81 rows (one row per mutually-exclusive time interval per subject) and 7 columns:

id Integer subject identifier (40 subjects, ids 1..40).

cluster Integer cluster identifier (8 clusters, 5 subjects each).

treatment Binary treatment indicator (0 or 1), balanced 20/20 across subjects. Treatment 1 has a higher illness hazard than treatment 0.

Tstart Numeric start time of the interval.

Tstop Numeric end time of the interval.

Sstart Integer state during the interval: 1 = Healthy, 2 = Ill, 3 = Dead (absorbing).

Sstop Integer state at `Tstop`, or equal to `Sstart` on a final, censored row.

Details

The transition structure is `trans_mat(list(c(2, 3), c(1, 3), integer(0)))`, i.e. Healthy <-> Ill, Healthy -> Dead, Ill -> Dead. Within each subject rows are temporally and spatially contiguous; censoring is encoded as `Sstart == Sstop` on the final row.

The same data are also shipped as a CSV at `system.file("extdata", "example_data.csv", package = "clusteredMSM")` for users who prefer to mimic loading their own file.

The generation script lives in `data-raw/example_msm.R` and is fully reproducible (`set.seed(2026)`).

Source

Simulated. See `data-raw/example_msm.R` in the package sources.

Examples

```
data(example_msm)
head(example_msm)
table(example_msm$Sstart, example_msm$Sstop)
```

`fit_chaz` *Fit Cox Model Stratified by Transition and Return Cumulative Hazards*

Description

Fits a Cox proportional hazards model stratified by transition type and returns the Breslow-estimated cumulative transition hazards in long format.

Usage

```
fit_chaz(data, tmat, weights = NULL)
```

Arguments

`data`	A data frame in long multistate format with columns `Tstart`, `Tstop`, `status`, and `trans`. Typically the output of `intervals_to_long()`.
`tmat`	A K x K transition matrix from `trans_mat()`. Used only to validate that all transition IDs in `data` are defined in `tmat`; not used in the fit itself.
`weights`	Optional numeric vector of length `nrow(data)` with observation weights. Pass `1 / data$clust.size` for the inverse-cluster-size-weighted estimator. `NULL` (default) means unweighted.

Details

The model fitted is `coxph(Surv(Tstart, Tstop, status) ~ strata(trans), method = "breslow")`. This is the Andersen-Gill counting-process formulation, which handles multiple intervals per subject – including non-monotone processes with recovery – provided the input data was correctly constructed (one row per sojourn x candidate transition).

Value

A data frame with columns:

- `time`: jump time of the transition.
- `Haz`: Breslow estimate of the cumulative transition hazard at `time`.
- `trans`: integer transition ID (matches `tmat`).

Sorted by `trans` then `time`.

Examples

```
data(example_msm)
tmat <- trans_mat(list(c(2, 3), c(1, 3), integer(0)),
                  names = c("Healthy", "Ill", "Dead"))

# fit_chaz takes long-format data; intervals_to_long() converts
# the interval format users supply. Most users reach fit_chaz
# indirectly via patp() -- this example shows the manual pipeline.
long <- intervals_to_long(example_msm, tmat)
haz  <- fit_chaz(long, tmat)
head(haz)
```

`intervals_to_long` *Convert Validated Interval Data to Long Multistate Format*

Description

Expands each row of an interval-format multistate dataset into one row per allowed transition out of the interval's starting state, with `status = 1` on the row whose destination matches the actual transition (and `0` on the others). The result is the long format expected by `fit_chaz` and other counting-process style routines.

Usage

```
intervals_to_long(data, tmat)
```

Arguments

`data`	A validated interval data frame with columns `id`, `Tstart`, `Tstop`, `Sstart`, `Sstop`, plus optional `cluster` and `group` columns. Run `validate_intervals()` first to confirm the data satisfy the package's contiguity and absorbing-state rules.
`tmat`	A K x K transition matrix from `trans_mat()`.

Details

Most users do not need to call this function directly – `patp()` runs the full pipeline (parse, validate, expand, fit). It is exported so advanced users can compose the lower-level building blocks (`fit_chaz`, `prodint_AJ`, `cluster_boot`) into custom analyses.

Value

A data frame in long multistate format with columns `id`, `Tstart`, `Tstop`, `from`, `to`, `trans`, `status`, plus `cluster` and `group` if present in `data`.

Examples

```
data(example_msm)
tmat <- trans_mat(list(c(2, 3), c(1, 3), integer(0)),
                  names = c("Healthy", "Ill", "Dead"))
long <- intervals_to_long(example_msm, tmat)
head(long)
```

`ks_pvalue` *Two-Sample Kolmogorov-Smirnov-Type Test via Cluster Bootstrap*

Description

Computes a p-value for the null hypothesis that a transition probability curve is equal across two groups, based on the supremum of the absolute difference between group-specific Aalen-Johansen estimators and a cluster-bootstrap reference distribution.

Usage

```
ks_pvalue(diff_point, diff_boot, scale)
```

Arguments

`diff_point`	Numeric vector. Observed group difference of the point estimator ($\hat{P}_1 - \hat{P}_0$) on a fixed time grid.
`diff_boot`	Numeric matrix. Cluster bootstrap replicates of the group difference: rows = time points (matching `diff_point`), columns = replicates.
`scale`	Numeric scalar. Asymptotic scaling factor applied to both the observed sup-statistic and the bootstrap analogue. Use $\sqrt{n}$ (n = total clusters) for the dependent-groups design (Bakoyannis 2021, Theorem 3); use $\sqrt{n_1 n_2/(n_1+n_2)}$ for the cluster-randomized design (Bakoyannis & Bandyopadhyay 2022, Theorem 2).

Details

The test statistic is $T = c_n \sup_t |\hat{P}_1(t) - \hat{P}_0(t)|$ with $c_n =$ `scale`, and the null distribution is approximated by the bootstrap analogue applied to centered bootstrap differences. The empirical p-value is invariant to `scale` (it appears on both sides of the comparison); the role of `scale` is to put the reported statistic on the correct asymptotic scale per the relevant theorem.

This v0.1 implementation uses a unit weight: every time point on the grid contributes equally to the supremum. Bakoyannis (2021) Section 2.5 instead recommends a time-varying weight $W(t) = \prod_p Y_p(t) / \sum_p Y_p(t)$ (the harmonic mean of the per-group at-risk processes), which down-weights regions where one group's at-risk set is small and naturally tames tail instability of the difference estimator. A weighted variant following Bakoyannis (2021) Section 2.5 and Bakoyannis & Bandyopadhyay (2022) is planned for v0.2.

Value

A list with elements `statistic` (the observed sup-statistic, on the chosen scale) and `p.value`.

`msm` *Construct a Multistate Interval Object for Use in patp() Formulas*

Description

Packages four columns – interval start time, interval end time, starting state, and ending state – into a single object suitable for use as the response in a `patp` formula. Conceptually analogous to `Surv()` in survival analysis: the user calls it inside a formula (`msm(Tstart, Tstop, Sstart, Sstop) ~ ...`).

Usage

```
msm(Tstart, Tstop, Sstart, Sstop)
```

Arguments

`Tstart`	Numeric vector. Start time of each interval.
`Tstop`	Numeric vector. End time of each interval.
`Sstart`	Integer-valued numeric vector. State occupied at `Tstart`. States must be whole numbers; the matrix backing stores them as double (matrices have a single storage mode), but values are validated to be integer-valued.
`Sstop`	Integer-valued numeric vector. State occupied at `Tstop`. Same storage caveat as `Sstart`. If `Sstart == Sstop`, the row represents a censored interval; this is permitted only on the final row of a subject's record.

Details

Used inside a model formula passed to `patp`, e.g. `patp(msm(Tstart, Tstop, Sstart, Sstop) ~ 1, ...)`. The four arguments can be named anything in the user's data – the formula machinery looks them up in the supplied `data`.

Value

A four-column matrix of class `"msm"` with columns `Tstart`, `Tstop`, `Sstart`, `Sstop`.

See Also

`patp`, `validate_intervals`.

Examples

```
Tstart <- c(0,   1.5, 0,   0)
Tstop  <- c(1.5, 3.0, 2.0, 1.2)
Sstart <- c(1,   2,   1,   1)
Sstop  <- c(2,   3,   3,   1)   # last row censored (S unchanged)
obj <- msm(Tstart, Tstop, Sstart, Sstop)
head(obj)
```

patp *Population-Averaged Transition Probabilities for Multistate Process Data*

Description

Computes the working-independence Aalen-Johansen estimator of the transition probability $P(X(t) = j \mid X(s) = h)$ for clustered or independent multistate process data, with cluster-bootstrap standard errors, pointwise confidence intervals, and (optionally) simultaneous confidence bands. If a grouping variable is supplied on the right-hand side of the formula, also conducts a two-sample Kolmogorov-Smirnov-type test of curve equality.

Usage

```
patp(
  formula,
  data,
  tmat,
  id,
  cluster = NA,
  h,
  j,
  s = 0,
  weighted = FALSE,
  LMAJ = FALSE,
  B = 1000,
  cband = FALSE,
  design = c("auto", "shared", "cluster_random", "indep_random"),
  level = 0.95,
  seed = NULL
)
```

Arguments

formula	A formula of the form `msm(Tstart, Tstop, Sstart, Sstop) ~ 1` for a one-sample analysis, or `msm(Tstart, Tstop, Sstart, Sstop) ~ group` for a two-sample analysis (estimate + test).
data	A data frame containing the variables in `formula` plus the `id` (and optional `cluster`) columns.
tmat	A K x K transition matrix, typically built with `trans_mat()`.
id	Character. Name of the subject ID column. Required.
cluster	Character or `NA`. Name of the cluster ID column. When `NA` (default), the bootstrap resamples individuals; when supplied, it resamples whole clusters.
h	Integer in 1..K. Starting state.
j	Integer in 1..K. Ending state.

s	Numeric scalar. Conditioning time. Default 0.
weighted	Logical. Inverse-cluster-size weighting to account for potentially informative cluster size (requires cluster). Default FALSE.
LMAJ	Logical. Use the landmark Aalen-Johansen estimator (recommended when s > 0 and the Markov assumption is questionable). Default FALSE.
B	Integer. Number of bootstrap replications. B = 0 skips inference and is permitted only for one-sample formulas. Default 1000.
cband	Logical. If TRUE, return simultaneous confidence band limits at the level set by level (recommended B >= 1000).
design	Character, two-sample only. One of "auto" (default), "shared", "cluster_random", "indep_random". Selects the two-sample regime; see the *Two-sample designs* section. "auto" infers the regime from the cluster/group structure of the data, defaulting to "indep_random" when each cluster carries a single group (a safer default than assuming cluster randomization). When that fallback fires, a warning is emitted asking the user to set "cluster_random" explicitly if the per-group cluster counts were fixed by cluster randomization.
level	Confidence level. Default 0.95.
seed	Optional integer for reproducible bootstrap.

Value

An S3 object of class patp containing:

- call, formula: the original call and formula.
- curves: a data frame with columns time, P, group (if two-sample), and (if B > 0) se, ll, ul – and ll.band, ul.band if cband = TRUE.
- test: NULL (one-sample) or a list with the observed K-S statistic and bootstrap p-value (two-sample).
- n_subjects, n_clusters, groups, h, j, s, B.

Two-sample designs

Three regimes are supported, each with its own bootstrap scheme and asymptotic scaling factor; the test statistic in every case is $T = c_n \sup_t |\hat{P}_1(t) - \hat{P}_0(t)|$.

"shared" **(case i): dependent groups (multicenter trial).** Every cluster contributes observations from both groups – e.g., each hospital randomizes its patients to treatment or control. Unstratified cluster bootstrap; $c_n = \sqrt{n}$ (Bakoyannis 2021, Theorem 3), with n the total number of clusters.

"cluster_random" **(case ii.a): cluster-randomized trial.** Each cluster is assigned to exactly one group, with per-group cluster counts n_1, n_2 fixed by the randomization. Stratified cluster bootstrap (resample within each group); $c_n = \sqrt{n_1 n_2/(n_1+n_2)}$ (Bakoyannis & Bandyopadhyay 2022, Theorem 2). This regime must be opted into; "auto" will not select it.

"indep_random" **(case ii.b): independent observational comparison.** Each cluster carries one group, but n_1, n_2 are random (the population, not the analyst, decided who landed in which group). Unstratified cluster bootstrap; $c_n = \sqrt{n_1 n_2/(n_1+n_2)}$. The asymptotic regime is the same

two-independent-samples limit as `"cluster_random"`; only the bootstrap differs. This is the `"auto"` default when each cluster carries a single group.

Mixed structures (some clusters carry both groups, some only one) are not supported in this version.

`patp()` validates the supplied data against the requested design and stops with an informative error if they disagree.

The K-S statistic in v0.1 uses unit weight at every time point. Weighted variants – following the harmonic-mean weight $W(t) = \prod_p Y_p(t) / \sum_p Y_p(t)$ of Bakoyannis (2021) Section 2.5 and the related construction in Bakoyannis & Bandyopadhyay (2022) – are planned for v0.2.

Standard errors and confidence intervals

The cluster bootstrap is run *once*, on the original probability scale, in line with Bakoyannis (2021) Theorem 2. The `se` column in `curves` is the bootstrap standard deviation of $\hat{P}(t)$ on that scale, i.e. `apply(boot_matrix, 1, sd, na.rm = TRUE)`.

Pointwise CIs (`ll`, `ul`) are then built on the cloglog scale $g(p) = \log(-\log p)$ using the delta-method standardization $\mathrm{SE}_g(t) = \mathrm{SE}(\hat{P}(t)) / |\hat{P}(t) \log \hat{P}(t)|$, and back-transformed via $p = \exp(-\exp(\cdot))$. Simultaneous bands (`ll.band`, `ul.band`) use the same SE_g; the `level`-quantile of the supremum gives the critical value.

Because the cloglog transformation is nonlinear, `se` and the resulting CI widths are *not* equal in general – the CI is asymmetric on the probability scale. Report `se` for descriptive purposes; use `ll`/`ul` for inference.

The simultaneous band is trimmed to the central 90% of observed jump times by default to avoid the band fanning out near the extremes of follow-up; outside that window `ll.band` and `ul.band` are `NA`. See `confidence_band()` for the `trim` argument.

Note: in this version the simultaneous band uses an equal-precision-type construction on the cloglog scale, which is asymptotically valid but differs from the Hall-Wellner-type construction of Bakoyannis (2021) Section 2.3 (planned for a future release). See `confidence_band()` for the full disclosure.

References

Bakoyannis, G. (2021). Nonparametric analysis of nonhomogeneous multistate processes with clustered observations. *Biometrics*, 77(2), 533-546. doi:10.1111/biom.13327

Bakoyannis, G., & Bandyopadhyay, D. (2022). Nonparametric tests for multistate processes with clustered data. *Annals of the Institute of Statistical Mathematics*, 74(5), 837-867. doi:10.1007/s1046302100819x

Putter H, Spitoni C (2018). Non-parametric estimation of transition probabilities in non-Markov multi-state models: the landmark Aalen-Johansen estimator. *Statistical Methods in Medical Research* 27(7):2081-2092. doi:10.1177/0962280216674497

See Also

`msm`, `trans_mat`, `validate_intervals`.

Examples

```
data(example_msm)
tmat <- trans_mat(list(c(2, 3), c(1, 3), integer(0)),
                  names = c("Healthy", "Ill", "Dead"))

# One-sample: P(Ill at t | Healthy at 0). B is small here for speed;
# use B >= 1000 for reported results.
fit <- patp(msm(Tstart, Tstop, Sstart, Sstop) ~ 1,
            data = example_msm, tmat = tmat,
            id = "id", cluster = "cluster",
            h = 1, j = 2, s = 0,
            B = 50, seed = 1)
fit

# Two-sample (estimate + test in one call). Each cluster in
# example_msm carries both treatment levels, so design = "shared".
tt <- patp(msm(Tstart, Tstop, Sstart, Sstop) ~ treatment,
           data = example_msm, tmat = tmat,
           id = "id", cluster = "cluster",
           h = 1, j = 2, B = 50,
           design = "shared", seed = 1)
tt
```

`print.patp` *Print Method for patp Objects*

Description

Concise summary of a `patp` fit. For two-sample fits, includes the K-S statistic and p-value.

Usage

```
## S3 method for class 'patp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	A `patp` object.
`digits`	Integer. Number of significant digits for printing.
`...`	Additional arguments (ignored).

Value

The object `x`, invisibly.

prodint_AJ *Aalen-Johansen Product Integral for General Multistate Processes*

Description

Computes the row of the transition probability matrix

$$P(s,t) = \prod_{u \in (s,t]} (I + dA(u))$$

starting from state h, given Nelson-Aalen transition hazard estimates.

Usage

```
prodint_AJ(haz_df, tmat, predt = 0, h = 1)
```

Arguments

haz_df	Data frame with columns time, Haz, trans, typically the output of fit_chaz() or the $Haz element of an mstate::msfit object. Haz is the cumulative (not incremental) hazard.
tmat	A K x K transition matrix from trans_mat(). Cyclic transitions (both [h, j] and [j, h] non-NA) are permitted.
predt	Numeric scalar. Starting time s. Probabilities are computed for t > predt. Default 0.
h	Integer in 1..K. Starting state.

Details

At each unique jump time u, the increment matrix $dA(u)$ has off-diagonal entry (h, j) equal to the Nelson-Aalen increment of the cumulative hazard for transition $h \to j$, and diagonal entries equal to the negative row sums. The transition probability matrix is then updated by $P \leftarrow P(I + dA(u))$.

Computational complexity is O(J K^2) where J is the number of unique jump times and K is the number of states.

Value

A data frame with columns time, pstate1, ..., pstateK. Row k gives $P(predt, time_k)[h,]$, the probabilities of being in each state at time_k given the process was in state h at predt.

Examples

```
data(example_msm)
tmat <- trans_mat(list(c(2, 3), c(1, 3), integer(0)),
                  names = c("Healthy", "Ill", "Dead"))

long <- intervals_to_long(example_msm, tmat)
```

```
haz  <- fit_chaz(long, tmat)
P    <- prodint_AJ(haz, tmat, predt = 0, h = 1)
head(P)
```

`state_at` *Identify Each Subject's State at a Given Time*

Description

Given a long-format multistate dataset, returns the state each subject occupies at time `s`. Used by the landmark Aalen-Johansen estimator to identify the risk set at a conditioning time.

Usage

```
state_at(data, s, id = "id")
```

Arguments

`data`	A data frame in long format with columns `Tstart`, `Tstop`, `from`, `to`, `status`, and the subject ID column named by `id`.
`s`	Numeric scalar. The time at which to evaluate states.
`id`	Character. Name of the subject ID column. Default `"id"`.

Details

A subject's state at time `s` is the `from` state of the interval that contains `s` (i.e., `Tstart <= s < Tstop`). Because the long format duplicates intervals across competing transitions, the result is deduplicated to one row per subject.

For exact ties (`s == Tstop`), the subject is treated as having transitioned: they appear in their new state if a subsequent interval exists, otherwise they are excluded as absorbed/censored.

Value

A data frame with columns `<id>` and `state`, one row per subject who is under observation at time `s`. Subjects whose follow-up has not yet started (`min(Tstart) > s`) or who have already entered an absorbing state (`max(Tstop) <= s` with `status == 1`) are excluded.

References

Putter H, Spitoni C (2018). Non-parametric estimation of transition probabilities in non-Markov multi-state models: the landmark Aalen-Johansen estimator. *Statistical Methods in Medical Research* 27(7):2081-2092. doi:10.1177/0962280216674497

Examples

```
data(example_msm)
tmat <- trans_mat(list(c(2, 3), c(1, 3), integer(0)),
                  names = c("Healthy", "Ill", "Dead"))
long <- intervals_to_long(example_msm, tmat)

# Each subject's state at the landmark time s = 1.5
head(state_at(long, s = 1.5, id = "id"))
```

`summary.patp` *Summary Method for patp Objects*

Description

Returns the full curve(s) and (if applicable) the test result.

Usage

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

Arguments

`object`	A `patp` object.
`...`	Additional arguments (ignored).

Value

A list with named components for further inspection.

`trans_mat` *Build a Transition Matrix for a Multistate Process*

Description

Constructs a square transition matrix in the format expected by `patp` and `prodint_AJ`. It supports cyclic transitions (e.g., illness-death with recovery).

Usage

```
trans_mat(x, names = NULL)
```

Arguments

x	A list of length K. Element k is an integer vector of states reachable directly from state k. Use integer(0) or NULL for absorbing states.
names	Optional character vector of length K with state names. If NULL, states are named "1", "2", ..., "K".

Value

A K x K integer matrix. Entry [h, j] is the integer transition ID for the direct transition h -> j, or NA if no such transition exists. Transition IDs are assigned in row-major order (all transitions out of state 1 first, then state 2, etc.). The matrix has dimnames list(from = names, to = names).

Examples

```
# Illness-death without recovery (progressive)
trans_mat(list(c(2, 3), 3, integer(0)),
          names = c("Healthy", "Ill", "Dead"))

# Illness-death WITH recovery (non-monotone)
trans_mat(list(c(2, 3), c(1, 3), integer(0)),
          names = c("Healthy", "Ill", "Dead"))

# Four-state multistate model with recovery from two intermediate states
trans_mat(list(c(2, 3, 4), c(1, 4), c(1, 4), integer(0)))
```

Index

∗ **datasets**
example_msm, 7

ci_cloglog, 2
cluster_boot, 3, *9*
confidence_band, 5, *14*
cut_at_lm, 6

example_msm, 7

fit_chaz, 8, *9*

intervals_to_long, *6*, *8*, 9

ks_pvalue, 10

msm, 11, *14*

patp, *9*, *11*, 12, *15*, *18*
print.patp, 15
prodint_AJ, *9*, 16, *18*

state_at, 17
summary.patp, 18
Surv, *11*

trans_mat, *9*, *12*, *14*, 18

validate_intervals, *9*, *11*, *14*