Package 'survPhenoStab'

November 17, 2026

Type Package

Title Stability-Driven Hierarchical Clustering for Multi-Endpoint Survival Phenotyping

Version 0.1.0

Description Implements the stability-driven hierarchical clustering framework for multi-endpoint patient phenotyping described in Roy Chowdhury et al., ``A stability-driven hierarchical clustering framework for multi-endpoint patient phenotyping in oncology'' (submitted to Computers in Biology and Medicine). Provides survival feature-space embedding for multiple time-to-event endpoints, a competing-risks data-generating mechanism for simulation, an exhaustive sweep over valid distance-linkage-K configurations, bootstrap Jaccard cluster-stability assessment, a chaining-artefact diagnostic (Delta_CA), cluster-number selection combining silhouette width, the gap statistic and bootstrap stability, and benchmarking against k-means and Gaussian mixture models.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0), mclust

Imports stats, graphics, grDevices, cluster, survival, rmarkdown, utils

Suggests testthat (>= 3.0.0), knitr, markdown

RoxygenNote 7.3.1

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Oindrila Roy Chowdhury [aut, cre],
Bapan Kalita [aut],
Dilip Chandra Nath [aut],
Atanu Bhattacharjee [aut]

Maintainer Oindrila Roy Chowdhury <oindrila6176@gmail.com>

Repository CRAN

Date/Publication 2026-11-17 17:00:02 UTC

Contents

benchmark_methods . . . 2
boot_jaccard . . . 3
chaining_diagnostic . . . 5
cluster_assign . . . 6
cluster_configs . . . 6
colon_survival . . . 7
cosine_dist . . . 8
embed_survival . . . 8
generate_report . . . 9
logrank_test . . . 11
mc_replicate . . . 11
plot_covariate_heatmap . . . 12
plot_dendrogram . . . 14
plot_km_by_cluster . . . 15
plot_silhouette . . . 15
plot_stability_heatmap . . . 16
select_k . . . 17
silhouette_values . . . 19
silhouette_width . . . 19
sim_competing_risks . . . 20

Index **22**

`benchmark_methods` *Benchmark hierarchical clustering against k-means and Gaussian mixtures*

Description

Table 6 in the manuscript reports these numbers but there was no code for k-means/GMM anywhere in the original scripts, so this rebuilds the comparison from scratch: the chosen hierarchical configuration against k-means, a Gaussian mixture (EM, full covariance), and Ward-Euclidean, all on the same embedding at the same K. Same three metrics as elsewhere in the package (J_min, ARI, silhouette) so the rows are directly comparable, plus wall-clock time since the manuscript reports that too (k-means/GMM are noticeably slower to bootstrap than hierarchical clustering, since each resample refits the whole model).

Usage

```
benchmark_methods(
  Z,
  k,
  proposed_distance = "cosine",
  proposed_linkage = "average",
  include_ward = TRUE,
  B = 200,
  kmeans_nstart_ref = 20,
  kmeans_nstart_boot = 3,
  seed = 42
)
```

Arguments

Z	numeric embedding matrix.
k	number of clusters.
proposed_distance, proposed_linkage	the configuration treated as "proposed" (default "cosine"/"average", the manuscript's recommended configuration); its partition is the ARI reference for every other method, and it is itself included as the first row of the result.
include_ward	logical; also benchmark Ward-Euclidean as an explicit baseline row (default TRUE).
B	bootstrap resamples for J_{min} (k-means and GMM require refitting on every resample, so this dominates runtime).
kmeans_nstart_ref, kmeans_nstart_boot	number of random k-means initialisations on the reference fit and on each bootstrap fit, respectively (defaults 20 and 3, matching the manuscript's stated computational protocol in Section 5).
seed	random seed.

Value

a data.frame with one row per method and columns method, J_min, ARI, silhouette, time_s.

---

boot_jaccard *Bootstrap Jaccard cluster-stability assessment*

Description

Ports jmin_for_config() / jmin_real(): assesses the stability of a hierarchical clustering configuration via bootstrap resampling (manuscript Section 2.5.2, Eq. 4). For each of B bootstrap resamples, the data are re-clustered under the same configuration and the per-cluster Jaccard coefficient against the reference partition is recorded; J_min is the minimum, over clusters, of the mean Jaccard across resamples.

Usage

```
boot_jaccard(
  Z,
  distance = c("cosine", "euclidean", "manhattan"),
  linkage = c("average", "complete", "ward.D2", "single"),
  k,
  B = 500,
  seed = 42
)
```

Arguments

`Z`	numeric embedding matrix (observations in rows).
`distance`, `linkage`, `k`	as in `cluster_assign`.
`B`	number of bootstrap resamples (default 500, the manuscript's primary-replicate value; the manuscript also uses B = 200 for the Monte Carlo replication and real-data analyses).
`seed`	random seed for the bootstrap resampling.

Value

a list of class `"spstab_config"` with elements:

`partition`	the reference-partition cluster labels
`J_min`	minimum mean per-cluster Jaccard
`J_min_sd`	bootstrap SD of the Jaccard for the limiting cluster
`J_per_cluster`	mean Jaccard for every cluster
`sizes`	reference cluster sizes
`stability_class`	`"stable"` (J_min > 0.75), `"borderline"` (0.50 < J_min <= 0.75) or `"unstable"` (J_min <= 0.50), following Hennig's convention (manuscript Section 2.5.2).
`ci`	a length-2 vector, the 2.5 interval of the per-resample minimum-cluster Jaccard.

Examples

```
set.seed(1)
Z <- matrix(rnorm(200), 100, 2)
res <- boot_jaccard(Z, distance = "euclidean", linkage = "ward.D2", k = 3, B = 50)
res$J_min
```

`chaining_diagnostic` *Chaining diagnostic: Delta_CA = ARI - J_min (manuscript Eq. 5, Section 2.5.4)*

Description

This one had no code behind it anywhere in the original analysis – the manuscript reports Delta_CA values in Table 2 but the formula was never actually implemented. Wrote it here directly from the definition.

Usage

```
chaining_diagnostic(
  Z,
  distance = c("cosine", "euclidean", "manhattan"),
  linkage = c("average", "complete", "ward.D2", "single"),
  k,
  B = 200,
  seed = 42,
  threshold = 0.2
)
```

Arguments

`Z`	numeric embedding matrix.
`distance, linkage, k`	as in `cluster_assign`.
`B`	bootstrap resamples (used for both J_min and ARI – see Details below on why they share the same resamples).
`seed`	random seed.
`threshold`	Delta_CA cutoff for flagging, default 0.20 (manuscript's value).

Details

The idea: single linkage tends to produce one big "everything" cluster plus a few near-singleton leftovers. Resample the data and re-cluster – the big cluster comes back looking almost identical every time (ARI stays high), but the tiny leftover clusters get reshuffled at random (J_min drops). ARI and J_min normally track each other; when they pull apart like this, that gap IS the chaining signature. Delta_CA just measures the gap directly instead of eyeballing two separate numbers.

Threshold of 0.20 is the manuscript's own cutoff – empirically chosen there, not derived from anything, so treat it as a rule of thumb rather than a hard boundary.

Value

list with `ARI`, `J_min`, `delta_CA`, `flagged`, and `stability` (the full `boot_jaccard` output, in case you want the per-cluster breakdown too).

cluster_assign *Cluster and assign labels for one distance-linkage-K configuration*

Description

Applies agglomerative hierarchical clustering under a given distance metric and linkage criterion and cuts the dendrogram at `K` clusters. Ward's criterion is restricted to Euclidean distance, matching the manuscript's stated restriction (Section 2.4) that Ward.D2 is not combined with non-Euclidean metrics.

Usage

```
cluster_assign(
  Z,
  distance = c("cosine", "euclidean", "manhattan"),
  linkage = c("average", "complete", "ward.D2", "single"),
  k
)
```

Arguments

`Z`	numeric embedding matrix, observations in rows (typically the output of `embed_survival`).
`distance`	one of `"cosine"`, `"euclidean"`, `"manhattan"`.
`linkage`	one of `"average"`, `"complete"`, `"ward.D2"`, `"single"`. `"ward.D2"` is only valid with `distance = "euclidean"`.
`k`	number of clusters to cut the dendrogram into.

Value

an integer vector of cluster labels, length `nrow(Z)`.

cluster_configs *Enumerate the valid distance-linkage-K configurations*

Description

Reproduces the 27-configuration grid of the manuscript (Section 2.4, Figure 1): 3 distance metrics x 3 linkage criteria (Ward.D2, complete, average), with Ward.D2 restricted to Euclidean distance (9 valid distance-linkage pairs), each evaluated at every value of `k` (default `3:5`), for 27 candidate configurations by default. Single linkage is deliberately excluded from the default grid, matching the manuscript's definition of the primary 27-configuration sweep (Section 2.4); pass `linkages` including `"single"` explicitly to reproduce the separate single-linkage chaining-artefact exploration of Table 2.

Usage

```
cluster_configs(
  distances = c("cosine", "euclidean", "manhattan"),
  linkages = c("average", "complete", "ward.D2"),
  k = 3:5
)
```

Arguments

distances	character vector of distance metrics to include.
linkages	character vector of linkage criteria to include. Default `c("average", "complete", "ward.D2")`, matching the manuscript's primary grid.
k	integer vector of cluster counts to evaluate (default `3:5`, matching the manuscript).

Value

a data.frame with columns `distance`, `linkage`, `k`, one row per valid configuration.

---

colon_survival *Colon cancer adjuvant chemotherapy dataset (wide, multi-endpoint form)*

Description

The public colon cancer dataset (Moertel et al. 1990), reshaped from the long form in the **survival** package (`survival::colon`) into one row per patient with both time-to-event endpoints as columns, ready for `embed_survival`. Used for the real-data application in the manuscript, Section 4.1.

Usage

```
colon_survival
```

Format

A data.frame with 929 rows and 5 columns:

id patient identifier

T_recur time to recurrence or censoring, in days

E_recur recurrence event indicator (1 = recurrence, 0 = censored)

T_death time to death or censoring, in days

E_death death event indicator (1 = death, 0 = censored)

Source

`survival::colon`; Moertel CG, Fleming TR, MacDonald JS, et al. (1990) "Levamisole and fluorouracil for adjuvant therapy of resected colon carcinoma." *New England Journal of Medicine* 322:352-358.

Examples

```
data(colon_survival)
Z <- embed_survival(
  time  = colon_survival[, c("T_recur", "T_death")],
  event = colon_survival[, c("E_recur", "E_death")]
)
dim(Z)
```

`cosine_dist` *Cosine distance matrix*

Description

Computes pairwise cosine distance (1 - cosine similarity) on a numeric matrix, guarding against zero-norm rows exactly as described in the manuscript (Section 2.3): row norms below `tol` are floored to `tol` before normalisation, and a warning is issued (matching the code's automatic near-zero-norm check).

Usage

```
cosine_dist(Z, tol = 1e-08)
```

Arguments

`Z`	a numeric matrix, observations in rows.
`tol`	minimum row norm; rows with $\|Z_i\| < tol$ are flagged.

Value

an object of class `"dist"` (as produced by `as.dist`), suitable for `stats::hclust()`.

`embed_survival` *Build a multi-endpoint survival feature-space embedding*

Description

Constructs the standardised, mixed continuous-binary feature-space embedding used throughout the stability-driven clustering framework (manuscript Section 2.2, Eq. 2): each time-to-event endpoint contributes a log-transformed event time and a binary event indicator, and the full matrix is centred and scaled to zero mean / unit variance per column.

Usage

```
embed_survival(time, event, log_floor = 0.001, standardize = TRUE)
```

Arguments

time	a numeric matrix or data.frame of observed times, one column per endpoint (e.g. columns OS, PFS, LRFS).
event	a numeric/integer matrix or data.frame of event indicators (1 = event, 0 = censored), same dimensions and column order as time.
log_floor	small positive constant used in log(pmax(time,log_floor)) to avoid -Inf for zero or near-zero times. Defaults to 1e-3, matching the simulation code.
standardize	logical; if TRUE (default) each column of the resulting embedding is centred and scaled to unit variance.

Details

When standardize = TRUE, a column with zero variance (for example the event indicator of an endpoint with no censoring, where every patient has the event) carries no information for clustering. Such columns are centred to zero rather than divided by a zero standard deviation, so the embedding contains no NaN values, and a warning names the affected columns.

Zero-norm rows (all-zero after standardisation) are not possible after centring in general, but if cosine_dist is used downstream on an embedding where every observation is identical in a subset of columns, near-zero-norm rows can occur; cosine_dist guards against this exactly as in the manuscript (Section 2.3).

Value

A numeric matrix with 2 * ncol(time) columns, ordered log T_1, E_1, log T_2, E_2, ..., with column names logT_<endpoint> / E_<endpoint> taken from colnames(time). A leading T_ prefix on the time column names (e.g. T_recur) is dropped, giving logT_recur / E_recur rather than logT_T_recur / E_T_recur.

Examples

```
set.seed(1)
time_mat  <- cbind(OS = rexp(50, 0.05), PFS = rexp(50, 0.08))
event_mat <- cbind(OS = rbinom(50, 1, 0.7), PFS = rbinom(50, 1, 0.8))
Z <- embed_survival(time_mat, event_mat)
dim(Z)
```

---

generate_report *Generate a self-contained HTML analysis report*

Description

Runs the stability-driven clustering pipeline on an embedding and knits a single self-contained HTML report covering: the bootstrap stability result for the chosen configuration, the stability heatmap across all valid configurations, the chaining diagnostic, a silhouette plot, a dendrogram, and (if supplied) a covariate heatmap and Kaplan-Meier curves by cluster. Intended as a quick, shareable summary of one analysis run, e.g. to attach to a manuscript revision or share with co-authors.

Usage

```
generate_report(
  Z,
  time = NULL,
  event = NULL,
  covariates = NULL,
  covariate_names = NULL,
  distance = c("cosine", "euclidean", "manhattan"),
  linkage = c("average", "complete", "ward.D2", "single"),
  k = 3,
  B = 200,
  compare_ward = TRUE,
  seed = 42,
  title = "survPhenoStab analysis report",
  output_file = "survPhenoStab_report.html",
  output_dir = getwd(),
  open = interactive()
)
```

Arguments

Z	numeric embedding matrix (typically from embed_survival or sim_competing_risks).
time, event	optional numeric matrices of times/event indicators (one column per endpoint) to include Kaplan-Meier curves by cluster; both must be supplied together, or both left NULL.
covariates	optional numeric matrix of covariates (one row per observation, aligned with Z) to include a covariate heatmap.
covariate_names	optional display names for covariates' columns.
distance, linkage, k	the configuration to report on in detail (default "cosine"/"average"/3, the manuscript's recommended default).
B	bootstrap resamples for the headline stability result (the stability heatmap and chaining diagnostic use min(B, 100) to keep report generation fast).
compare_ward	logical; also report the Ward-Euclidean J_min for comparison (default TRUE), unless the selected configuration already *is* Ward-Euclidean.
seed	random seed.
title	report title.
output_file	file name for the rendered report (default "survPhenoStab_report.html").
output_dir	directory to write the report into (default the current working directory).
open	logical; if TRUE (default, interactive sessions only) open the rendered report in a browser after knitting.

Value

invisibly, the full path to the rendered HTML file.

Examples

```
sim <- sim_competing_risks(n = 200, seed = 1)
generate_report(sim$embedding, time = sim$time, event = sim$event,
                covariates = sim$covariates, B = 50,
                output_dir = tempdir(), open = FALSE)
```

---

logrank_test *Multi-group log-rank test statistic*

Description

Direct port of `logrank_chisq()`: computes the multi-sample log-rank chi-squared statistic (observed-minus-expected events, with the standard log-rank variance-covariance matrix) for testing equality of survival across the groups defined by `group` (e.g. recovered cluster labels). Used descriptively in the manuscript (Section 3.3 / 4.1) to characterise separation between recovered phenotypes; as noted there, because cluster assignment and this test use the same data, the resulting p-value is descriptive only and does not carry standard inferential interpretation.

Usage

```
logrank_test(time, event, group)
```

Arguments

`time`	numeric vector of observed times.
`event`	integer/logical vector of event indicators (1/TRUE = event).
`group`	vector (numeric, character or factor) of group labels, same length as `time`.

Value

a list with `chisq`, `df`, and `p_value`.

---

mc_replicate *Monte Carlo replication of the cosine-average vs Ward-Euclidean comparison*

Description

Ports `one_replicate()` plus the driving loop from `run_mc_python.py`: repeats data generation, embedding, clustering and bootstrap-stability validation across `M` independent simulation replicates (manuscript Section 2.6), recording the per-replicate difference $\Delta J_{min}^{(m)}$ (Eq. 6) and testing whether it is consistently positive via a paired one-sided Wilcoxon signed-rank test.

Usage

```
mc_replicate(
  M = 200,
  n = 518,
  B = 200,
  base_seed = 1000,
  sim_args = list(),
  progress = TRUE
)
```

Arguments

M	number of Monte Carlo replicates (default 200, matching the manuscript).
n	patients per replicate (default 518).
B	bootstrap resamples per replicate per configuration (default 200, matching the manuscript's Monte Carlo protocol; the manuscript uses B = 500 only for the single primary-replicate analysis).
base_seed	base random seed; replicate m uses seed base_seed + m.
sim_args	a list of additional arguments passed to sim_competing_risks (e.g. rho, tau, eta, params) for sensitivity analyses (manuscript Table 4).
progress	logical; print progress every 10 replicates.

Value

an object of class "spstab_mc": a list with cos_avg, ward, delta (numeric vectors, length M), wilcoxon (the htest object, one-sided "greater"), and summary (a one-row data.frame of means, SDs, percent positive).

Examples

```
mc <- mc_replicate(M = 20, n = 100, B = 50)
mc
```

---

plot_covariate_heatmap

Heatmap of standardised covariate profiles across clusters

Description

Reproduces the manuscript's Figure 4: for each cluster, shows the within-cluster mean of each (standardised) covariate as a coloured grid cell, so that clusters can be characterised by which covariates are above/below the overall cohort average.

Usage

```
plot_covariate_heatmap(
  covariates,
  labels,
  covariate_names = NULL,
  standardize = TRUE,
  order_by = NULL,
  main = "Standardised covariate profiles across clusters"
)
```

Arguments

covariates	a numeric matrix or data.frame of covariates, one row per patient, one column per covariate (e.g. the covariates element returned by sim_competing_risks, or any covariate matrix of your own aligned with labels).
labels	integer/factor vector of cluster labels, length nrow(covariates) (e.g. boot_jaccard(...)$partition).
covariate_names	optional character vector of display names for the covariate columns (default: colnames(covariates)).
standardize	logical; if TRUE (default) each covariate column is standardised to z-scores relative to the whole cohort before averaging within clusters, matching the manuscript's Figure 4.
order_by	optionally, a covariate name (or column index) to sort the displayed cluster rows by (descending), so e.g. the highest-risk cluster appears first. Default NULL keeps clusters in label order.
main	plot title.

Value

invisibly, the cluster x covariate matrix of means that was plotted (so you can inspect the exact numbers behind the colours).

Examples

```
sim <- sim_competing_risks(n = 200, seed = 1)
labels <- cluster_assign(sim$embedding, "cosine", "average", k = 3)
plot_covariate_heatmap(sim$covariates, labels,
  covariate_names = c("Functional status", "Disease severity",
                      "Comorbidity index", "Age", "Biomarker level"))
```

`plot_dendrogram` *Dendrogram of a hierarchical clustering configuration*

Description

Plots the merge tree for a given distance-linkage configuration and outlines the k-cluster cut with coloured boxes, using `stats::hclust` / `stats::rect.hclust` under the hood. Useful for seeing *how* patients get merged step by step, which a cluster-label vector alone doesn't show.

Usage

```
plot_dendrogram(
  Z,
  distance = c("cosine", "euclidean", "manhattan"),
  linkage = c("average", "complete", "ward.D2", "single"),
  k = 3,
  labels = NULL,
  main = NULL,
  box_border = c("#B2182B", "#2166AC", "#1B7837", "#B15928", "#762A83")
)
```

Arguments

`Z`	numeric embedding matrix (observations in rows).
`distance, linkage`	as in `cluster_assign`.
`k`	number of clusters to outline with boxes (set to `NULL` to just show the tree with no cut).
`labels`	optional vector of leaf labels (e.g. patient IDs); default suppresses leaf labels, which is usually more readable for n > ~50.
`main`	plot title.
`box_border`	colours used for the k cluster boxes (recycled if shorter than k).

Value

invisibly, the underlying `hclust` object.

Examples

```
sim <- sim_competing_risks(n = 80, seed = 1)
plot_dendrogram(sim$embedding, "cosine", "average", k = 3)
```

plot_km_by_cluster *Kaplan-Meier survival curves by recovered cluster, per endpoint*

Description

Reproduces the manuscript's Figure 3: for one or more time-to-event endpoints, plots Kaplan-Meier curves split by cluster label, with the descriptive log-rank chi-squared (via `logrank_test`) annotated on each panel. As in the manuscript, these log-rank statistics are descriptive only, since the same data were used to form the clusters and to test them.

Usage

```
plot_km_by_cluster(time, event, labels, endpoint_names = NULL, mfrow = NULL)
```

Arguments

`time`	a numeric matrix/data.frame of times, one column per endpoint (e.g. the `time` element from `sim_competing_risks`, or `colon_survival[, c("T_recur","T_death")]`).
`event`	a numeric/integer matrix/data.frame of event indicators, same shape as `time`.
`labels`	cluster labels, length `nrow(time)`.
`endpoint_names`	optional display names for the endpoints (default: `colnames(time)`).
`mfrow`	optional `c(nrow, ncol)` panel layout; default lays panels out in one row.

Value

invisibly, a list of `logrank_test` results, one per endpoint.

Examples

```
sim <- sim_competing_risks(n = 200, seed = 1)
labels <- cluster_assign(sim$embedding, "cosine", "average", k = 3)
plot_km_by_cluster(sim$time, sim$event, labels)
```

plot_silhouette *Per-observation silhouette plot*

Description

The standard silhouette diagnostic plot (Rousseeuw 1987): one horizontal bar per patient, grouped by cluster and sorted from best- to worst-clustered within each group, so you can see which *individual* patients are ambiguous or mis-clustered rather than only the cluster average (as reported by `silhouette_width`).

Usage

```
plot_silhouette(
  Z,
  labels,
  distance = c("cosine", "euclidean", "manhattan"),
  main = NULL
)
```

Arguments

Z	numeric embedding matrix (observations in rows).
labels	integer/factor vector of cluster labels.
distance	one of "cosine", "euclidean", "manhattan".
main	plot title.

Value

invisibly, a data.frame with columns label, sil (silhouette value) and order (plotting order), one row per observation.

Examples

```
sim <- sim_competing_risks(n = 150, seed = 1)
labs <- cluster_assign(sim$embedding, "cosine", "average", k = 3)
plot_silhouette(sim$embedding, labs, "cosine")
```

---

plot_stability_heatmap

Heatmap of bootstrap stability (J_min) across distance-linkage configurations

---

Description

New visualisation (not in the manuscript): computes J_min via boot_jaccard for every valid distance-linkage combination (see cluster_configs) at a fixed k, and displays the result as a coloured grid. This makes the manuscript's central argument – "check stability empirically rather than assuming a default" – visible at a glance, instead of requiring a table lookup.

Usage

```
plot_stability_heatmap(
  Z,
  k,
  distances = c("cosine", "euclidean", "manhattan"),
  linkages = c("average", "complete", "ward.D2"),
```

```
  B = 100,
  seed = 42,
  main = sprintf("Bootstrap stability (J_min) at K = %d", k)
)
```

Arguments

Z	numeric embedding matrix.
k	number of clusters to evaluate at.
distances, linkages	passed to cluster_configs.
B	bootstrap resamples per configuration (kept modest by default since this function fits length(distances) x length(linkages) configurations).
seed	random seed.
main	plot title.

Value

invisibly, a distance x linkage matrix of J_min values (NA where a combination is invalid, e.g. Ward.D2 with a non-Euclidean distance).

Examples

```
sim <- sim_competing_risks(n = 150, seed = 1)
plot_stability_heatmap(sim$embedding, k = 3, B = 30)
```

---

select_k *Pick K by combining silhouette, gap statistic and bootstrap stability*

Description

Another one with no existing code – Section 2.5.5 describes the procedure in prose (three criteria, majority vote) but nothing in the original scripts actually ran it.

Usage

```
select_k(
  Z,
  distance = c("cosine", "euclidean", "manhattan"),
  linkage = c("average", "complete", "ward.D2", "single"),
  k_range = 2:6,
  B = 200,
  gap_B = 100,
  min_frac = 0.05,
  seed = 42
)
```

Arguments

Z	numeric embedding matrix.
distance, linkage	as in cluster_assign.
k_range	integer vector of candidate K values (default 2:6, matching the manuscript's Section 3.2 sweep).
B	number of bootstrap resamples for the stability curve (passed to boot_jaccard).
gap_B	number of Monte Carlo replicates used by cluster::clusGap for the gap statistic's reference distribution.
min_frac	minimum cluster size as a fraction of n (default 0.05).
seed	random seed.

Details

Why three criteria instead of one: silhouette width and the gap statistic are both "separation" measures – they ask whether the clusters look distinct from each other on a single fixed dataset. They say nothing about whether the same clusters would show up again if you had collected a slightly different sample. That's what the bootstrap J_min curve adds. A K that only wins on separation but not stability is probably overfitting to this one sample, so requiring 2-out-of-3 agreement is a guard against that rather than just an arbitrary voting rule.

The minimum-cluster-size rule exists because of a real failure mode you can hit with bootstrap stability alone: pushing K higher and higher eventually peels off tiny 1-2 patient clusters that look artificially "stable" in the bootstrap (small clusters can spuriously resample well just because there's so little of them to get wrong). Excluding any K whose smallest cluster is under 5 stops that from winning the vote.

Worth knowing: this can return k_selected = NA if none of the three criteria agree. That's not a bug – it's the function honestly reporting "there's no clean consensus here," which does happen on real (messy) data. See votes to see how close it was.

Value

a list with elements:

silhouette	named numeric vector, average silhouette width per K
gap	the clusGap object
k_silhouette	K selected by maximum silhouette width
k_gap	K selected by the gap statistic's one-SE rule
J_min_curve	named numeric vector, bootstrap J_min per K, after excluding K values that violate the minimum cluster size constraint
k_stability	K maximising the (constrained) stability curve
k_selected	final K, i.e. the value in k_range chosen by at least two of the three criteria above, or NA if no such K exists (see votes)
votes	table of how many of the three criteria selected each K

silhouette_values *Per-observation silhouette values under a given distance metric*

Description

Computes the silhouette value (manuscript Section 2.5.1, Eq. 3) for every observation individually, directly on the distance matrix implied by `distance`. `silhouette_width` is a thin wrapper that averages this vector; use this function directly when you need per-patient values, e.g. for `plot_silhouette`.

Usage

```
silhouette_values(Z, labels, distance = c("cosine", "euclidean", "manhattan"))
```

Arguments

`Z`	numeric embedding matrix (observations in rows).
`labels`	integer vector of cluster labels, length `nrow(Z)`.
`distance`	one of `"cosine"`, `"euclidean"`, `"manhattan"`.

Value

a numeric vector of length `nrow(Z)`, one silhouette value per observation, in the same order as `labels`.

silhouette_width *Average silhouette width under a given distance metric*

Description

Ports `silhouette_score()`: the mean of `silhouette_values` (manuscript Section 2.5.1, Eq. 3). As noted in the manuscript, silhouette values are only comparable *within* a distance metric family, not across metrics.

Usage

```
silhouette_width(Z, labels, distance = c("cosine", "euclidean", "manhattan"))
```

Arguments

`Z`	numeric embedding matrix (observations in rows).
`labels`	integer vector of cluster labels, length `nrow(Z)`.
`distance`	one of `"cosine"`, `"euclidean"`, `"manhattan"`.

Value

the average silhouette width (numeric scalar).

`sim_competing_risks` *Simulate multi-endpoint competing-risks survival data*

Description

Ports the competing-risks data-generating mechanism used in the manuscript (Section 2.1, Eq. 1): three correlated time-to-event endpoints (OS, PFS, LRFS) are generated from Weibull cause-specific hazards with correlated Gaussian covariates, administrative censoring at `tau`, random exponential censoring, and the clinical constraint T_PFS <= T_OS enforced by rejection sampling.

Usage

```
sim_competing_risks(
  n = 518,
  seed = 2024,
  rho = 0.3,
  tau = 60,
  eta = 0.015,
 params = list(OS = list(lam = 0.01, gam = 1.4, beta = c(0.45, 0.3, -0.2, 0.25, -0.15)),
  PFS = list(lam = 0.018, gam = 1.25, beta = c(0.35, 0.4, -0.25, 0.2, -0.1)), LRFS =
    list(lam = 0.012, gam = 1.3, beta = c(0.3, 0.2, -0.15, 0.35, -0.2))),
  max_reject_iter = 50
)
```

Arguments

`n`	number of patients to simulate. Default 518 (primary simulation replicate in the manuscript).
`seed`	integer random seed for reproducibility.
`rho`	constant pairwise covariate correlation (default 0.3).
`tau`	administrative censoring time in months (default 60).
`eta`	rate of the random exponential censoring distribution (default 0.015), calibrated for approximately 30 `tau = 60` jointly with the default `params`.
`params`	a named list with one element per endpoint (default `"OS"`, `"PFS"`, `"LRFS"`), each a list with elements `lam` (Weibull scale), `gam` (Weibull shape) and `beta` (numeric vector of covariate-effect coefficients). Defaults reproduce Table 1 of the manuscript (calibrated lambdas 0.010 / 0.018 / 0.012).
`max_reject_iter`	maximum rejection-sampling iterations used to enforce T_PFS <= T_OS before falling back to T_PFS <- T_OS for any remaining violations (default 50, matching the analysis code).

Details

The independence-assumption caveat in the manuscript (Section 2.3) applies here too: the three endpoints are not generated independently (they share covariates X and the T_PFS <= T_OS constraint), so downstream distance metrics that assume independent dimensions are applied under a violated assumption by construction.

Value

A list with elements:

`time`	n x n_endpoint matrix of observed times
`event`	n x n_endpoint matrix of event indicators
`covariates`	n x p matrix of the underlying Gaussian covariates
`embedding`	the standardised feature-space embedding, i.e. `embed_survival(time, event)`

Examples

```
sim <- sim_competing_risks(n = 100, seed = 2024)
dim(sim$embedding)
```

Index

∗ **datasets**
colon_survival, 7

as.dist, *8*

benchmark_methods, 2
boot_jaccard, 3, *5*, *16*, *18*

chaining_diagnostic, 5
cluster_assign, *4, 5*, 6, *14*, *18*
cluster_configs, 6, *16, 17*
colon_survival, 7
cosine_dist, 8

embed_survival, *6, 7*, 8, *10*

generate_report, 9

logrank_test, 11, *15*

mc_replicate, 11

plot_covariate_heatmap, 12
plot_dendrogram, 14
plot_km_by_cluster, 15
plot_silhouette, 15, *19*
plot_stability_heatmap, 16

select_k, 17
silhouette_values, *19*, 19
silhouette_width, *15*, *19*, 19
sim_competing_risks, *10*, *12, 13*, *15*, 20