

Package ‘unsurv’

September 17, 2026

Title Unsupervised Clustering of Individualized Survival Curves

Version 0.7.2

Author Imad El Badisy [aut, cre]

Maintainer Imad El Badisy <elbadisyimad@gmail.com>

Date 2026-08-31

Description Tools for clustering individualized survival curves using the Partitioning Around Medoids (PAM) algorithm, with monotonic enforcement, optional smoothing, weighted distances (L1/L2), automatic K selection via silhouette width, prediction for new curves, basic stability checks, and plotting helpers. The method is described in El Badisy (2026) <doi:10.1093/bioadv/vbag218>.

License MIT + file LICENSE

URL <https://CRAN.R-project.org/package=unsurv>

BugReports <https://github.com/ielbadisy/unsurv/issues>

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports stats, cluster, ggplot2

Suggests scales, survival, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-17 10:10:02 UTC

Contents

unsurv-package	2
autoplot.unsurv	3

autoplot.unsurv_compare	4
plot.unsurv	5
plot_stability	6
predict.unsurv	6
print.summary.unsurv	7
print.unsurv	8
print.unsurv_compare	9
summary.unsurv	9
unsurv	10
unsurv_compare	13
unsurv_stability	14
Index	16

unsurv-package	<i>unsurv: Unsupervised clustering of individualized survival curves</i>
----------------	--

Description

The **unsurv** package provides tools for unsupervised clustering of individualized survival curves using medoid-based clustering (PAM).

Details

It is designed for settings where each individual is represented by a survival probability curve evaluated on a common time grid, such as predictions from:

- Kaplan–Meier estimates,
- Cox models,
- parametric survival models,
- deep learning survival models,
- or other individualized survival predictors.

Core features include:

- PAM clustering using weighted L1 or L2 distances,
- automatic cluster selection via silhouette width,
- optional monotonicity enforcement,
- optional median smoothing,
- prediction of cluster membership for new curves,
- stability assessment via resampling and Adjusted Rand Index,
- base R and ggplot2 visualization methods.

Main functions:

- [unsurv](#) — fit clustering model

- `predict.unsurv` — predict cluster membership
- `plot.unsurv` — plot medoid curves
- `summary.unsurv` — summarize clustering
- `unsurv_stability` — assess stability

Author(s)

Imad EL BADISY

References

Kaufman, L., & Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

See Also

Useful links:

- <https://CRAN.R-project.org/package=unsurv>
- Report bugs at <https://github.com/ielbadisy/unsurv/issues>

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  n <- 10
  times <- seq(0, 5, length.out = 40)
  rates <- sample(c(0.2, 0.6), n, TRUE)
  S <- sapply(times, function(t) exp(-rates * t))

  fit <- unsurv(S, times, K = 2)
  plot(fit)
}
```

autoplot.unsurv

ggplot2 autoplot for unsurv objects

Description

ggplot2 autoplot for unsurv objects

Usage

```
## S3 method for class 'unsurv'
autoplot(object, ...)
```

Arguments

object	An object of class "unsurv".
...	Unused.

Value

A ggplot object.

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  times <- seq(0, 4, length.out = 25)
  grp <- rep(1:2, each = 10)
  rates <- c(0.2, 0.55)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  fit <- unsurv(S, times, K = 2)
  ggplot2::autoplot(fit)
}
```

autoplot.unsurv_compare

Kaplan-Meier plot for a partition comparison

Description

Plots Kaplan-Meier survival curves for each cluster, faceted by comparison method, for an object returned by [unsurv_compare](#).

Usage

```
## S3 method for class 'unsurv_compare'
autoplot(object, ...)
```

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

Arguments

object	An object of class "unsurv_compare".
...	Unused.
x	An object of class "unsurv_compare".

Value

A ggplot object.

Examples

```

if (requireNamespace("survival", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(1)
  n <- 120
  time <- stats::rexp(n, 0.1)
  status <- sample(0:1, n, TRUE)
  labs <- list(
    unsurv_curve = sample(1:3, n, TRUE),
    scalar_risk = sample(1:3, n, TRUE)
  )
  cmp <- unsurv_compare(labs, time, status)
  ggplot2::autoplot(cmp)
}

```

plot.unsurv

Plot medoid survival curves from an unsurv fit

Description

Produces a base R plot of the cluster medoid survival curves stored in the fitted object.

Usage

```

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

```

Arguments

x An object of class "unsurv".

... Additional arguments passed to [matplot](#) (e.g., lwd).

Value

Invisibly returns x.

Examples

```

if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  times <- seq(0, 4, length.out = 30)
  grp <- rep(1:2, each = 8)
  rates <- c(0.2, 0.55)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  fit <- unsurv(S, times, K = 2)
  plot(fit)
}

```

plot_stability	<i>Plot stability distribution</i>
----------------	------------------------------------

Description

Convenience helper to visualize the distribution of Adjusted Rand Index values returned by `unsurv_stability(...)` when `return_distribution = TRUE`.

Usage

```
plot_stability(stab)
```

Arguments

stab	Either the numeric vector of ARI values or the list returned by <code>unsurv_stability(..., return_distribution = TRUE)</code> .
------	--

Value

A ggplot histogram with a dashed line at the mean ARI.

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  times <- seq(0, 4, length.out = 25)
  grp <- rep(1:2, each = 10)
  rates <- c(0.2, 0.55)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  fit <- unsurv(S, times, K = 2)
  stab <- unsurv_stability(S, times, fit, B = 6, frac = 0.7)
  plot_stability(stab)
}
```

predict.unsurv	<i>Predict cluster membership for new survival curves</i>
----------------	---

Description

Assigns new survival-probability curves to clusters using the medoids from a fitted `unsurv` object. New curves are preprocessed using the same weighting, optional monotonic enforcement, smoothing, and standardization parameters as the fitted model.

Usage

```
## S3 method for class 'unsurv'
predict(object, newdata, clamp = TRUE, ...)
```

Arguments

object	An object of class "unsurv", returned by unsurv .
newdata	Numeric matrix of survival probabilities with shape $n_{new} \times m$, where columns correspond to the same time grid used during fitting.
clamp	Logical; if TRUE, clamps values to $[0, 1]$ before preprocessing.
...	Unused. Included for compatibility with the generic.

Details

Cluster assignment is performed by computing distances between the new curves and the stored medoid curves in the weighted feature space defined during fitting. The distance metric ("L1" or "L2") and any standardization parameters are reused from the fitted model.

Value

An integer vector of cluster labels of length `nrow(newdata)`, taking values in `1, ..., object$K`.

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  n <- 60; Q <- 40
  times <- seq(0, 5, length.out = Q)
  grp <- sample(1:2, n, TRUE)
  rates <- c(0.2, 0.6)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  S <- S + matrix(stats::rnorm(n * Q, 0, 0.02), nrow = n)

  fit <- unsurv(S, times, K = 2)

  # predict cluster membership for first 5 curves
  predict(fit, S[1:5, ])
}
```

`print.summary.unsurv` *Print a summary of an unsurv model*

Description

Print a summary of an unsurv model

Usage

```
## S3 method for class 'summary.unsurv'
print(x, ...)
```

Arguments

`x` An object of class "summary.unsurv".

`...` Unused.

Value

Invisibly returns `x`.

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  times <- seq(0, 4, length.out = 20)
  grp <- rep(1:2, each = 6)
  rates <- c(0.2, 0.6)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  fit <- unsurv(S, times, K = 2)
  s <- summary(fit)
  print(s)
}
```

```
print.unsurv
```

```
Print an unsurv model
```

Description

Print an unsurv model

Usage

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

Arguments

`x` An object of class "unsurv".

`...` Unused.

Value

Invisibly returns `x`.

Examples

```

if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  times <- seq(0, 4, length.out = 20)
  grp <- rep(1:2, each = 6)
  rates <- c(0.2, 0.6)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  fit <- unsurv(S, times, K = 2)
  print(fit)
}

```

```
print.unsurv_compare
```

Print a partition comparison

Description

Print a partition comparison

Usage

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

Arguments

x	An object of class "unsurv_compare".
...	Unused.

Value

Invisibly returns x.

```
summary.unsurv
```

Summarize an unsurv model

Description

Summarize an unsurv model

Usage

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

Arguments

object An object of class "unsurv".
 ... Unused.

Value

An object of class "summary.unsurv" with elements:

- K: number of clusters
- silhouette_mean: mean silhouette width
- size: cluster sizes

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  times <- seq(0, 4, length.out = 20)
  grp <- rep(1:2, each = 6)
  rates <- c(0.2, 0.6)
  S <- sapply(times, function(t) exp(-rates[grp] * t))
  fit <- unsurv(S, times, K = 2)
  summary(fit)
}
```

 unsurv

Unsupervised clustering of individualized survival curves

Description

Clusters individuals using their survival-probability curves evaluated on a common time grid. The method computes a weighted feature representation of the curves and applies PAM (Partitioning Around Medoids) on the resulting dissimilarity matrix. If K is not provided, it is selected by maximizing the mean silhouette width over $K = 2, \dots, K_{\max}$.

Fits an unsupervised clustering model on survival-probability curves evaluated on a common time grid. Clustering is performed using PAM (Partitioning Around Medoids) on a weighted feature representation of the curves.

Usage

```
unsurv(
  S,
  times,
  K = NULL,
  K_max = 10,
  distance = c("L2", "L1"),
  weights = NULL,
  enforce_monotone = TRUE,
```

```

    smooth_median_width = 0,
    standardize_cols = FALSE,
    eps_jitter = 0.001,
    seed = NULL
)

unsurv(
  S,
  times,
  K = NULL,
  K_max = 10,
  distance = c("L2", "L1"),
  weights = NULL,
  enforce_monotone = TRUE,
  smooth_median_width = 0,
  standardize_cols = FALSE,
  eps_jitter = 0.001,
  seed = NULL
)
```

Arguments

<code>S</code>	Numeric matrix of survival probabilities with shape $n \times m$. Rows are subjects, columns correspond to times. Values are clamped to $[0, 1]$.
<code>times</code>	Numeric vector of length m (strictly increasing time grid).
<code>K</code>	Optional integer number of clusters. If NULL, selected by silhouette.
<code>K_max</code>	Maximum K considered when K is NULL.
<code>distance</code>	Distance type: "L2" (euclidean) or "L1" (manhattan).
<code>weights</code>	Optional nonnegative vector of length m for time-point weights. If NULL, trapezoidal weights are used.
<code>enforce_monotone</code>	Logical; enforce non-increasing survival curves over time.
<code>smooth_median_width</code>	Integer; if ≥ 3 and odd, apply median smoothing along time.
<code>standardize_cols</code>	Logical; standardize feature columns before clustering.
<code>eps_jitter</code>	Nonnegative numeric; feature-space Gaussian jitter sd to break ties.
<code>seed</code>	Optional integer seed.

Details

This function requires the **cluster** package for PAM clustering and silhouette widths.

The returned object stores medoid curves and metadata required for prediction on new curves via [predict](#) (method `predict.unsurv`).

If K is NULL, the number of clusters is selected by maximizing the mean silhouette width over $K = 2, \dots, K_{\max}$.

Requires the **cluster** package (recommended in Suggests).

Value

An object of class "unsurv" with components including:

- clusters: integer vector of cluster assignments
- K: number of clusters
- times: time grid
- medoids: medoid survival curves (one per cluster)
- silhouette_mean: mean silhouette width
- plus preprocessing/settings fields used for prediction

An object of class "unsurv".

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(2025)
  n <- 40; Q <- 30
  times <- seq(0, 5, length.out = Q)
  rates <- c(0.12, 0.38, 0.8)
  grp <- sample(1:3, n, TRUE, c(0.4, 0.4, 0.2))
  S <- t(vapply(1:n, function(i)
    pmin(pmax(exp(-rates[grp[i]] * times) + rnorm(Q, 0, 0.01), 0), 1),
    numeric(Q)
  ))
  fit <- unsurv(S, times, K = NULL, K_max = 6, distance = "L2",
    enforce_monotone = TRUE, standardize_cols = FALSE,
    eps_jitter = 0, seed = NULL)
  print(fit)
  summary(fit)
  plot(fit)

  pred <- predict(fit, S[1:5, ])
  pred
}
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(1)
  n <- 40
  times <- seq(0, 5, length.out = 30)
  grp <- sample(1:2, n, TRUE)
  rates <- ifelse(grp == 1, 0.2, 0.6)
  S <- sapply(times, function(t) exp(-rates * t))
  S <- S + matrix(stats::rnorm(n * length(times), 0, 0.02), nrow = n)
  fit <- unsurv(S, times, K = NULL, K_max = 6, seed = 123)
  table(fit$clusters, grp)
}
```

unsurv_compare

*Compare cluster partitions against observed survival outcomes***Description**

Summarizes and compares one or more cluster-label partitions of the same individuals against observed time-to-event outcomes. This is intended for comparing an unsurv curve-based partition against baseline partitions (e.g., PAM on a scalar risk summary, or PAM on covariate PCA scores), or for comparing the same partitioning rule applied to different patient sets (e.g., a partition-defining set and a held-out validation set) to check that survival separation generalizes.

Usage

```
unsurv_compare(labels, time, status, reference = 1)
```

Arguments

labels	A named list of cluster-label vectors (integer or factor), each of the same length as time/status. Names are used as method labels; unnamed elements are labeled "method1", "method2", etc.
time	Numeric vector of observed follow-up times.
status	Numeric/integer vector of event indicators (1 = event, 0 = censored).
reference	Name or integer index of the element of labels used as the reference partition for Adjusted Rand Index (ARI) agreement. Defaults to the first element.

Details

Requires the **survival** package for Kaplan-Meier medians.

For each partition, the Adjusted Rand Index quantifies agreement with the reference partition. A log-rank test is deliberately not reported: when a partition is itself fit to separate the curves (as unsurv and the baselines are), a log-rank test against those same labels is circular and close to guaranteed to be "significant," so it is not a fair basis for comparing methods. Instead, per-cluster Kaplan-Meier medians are reported, which is useful for checking whether the ordering of clusters by survival (e.g., "cluster 2 has better survival than clusters 1 and 3") is preserved across sets, such as a partition-defining set and an independent validation set.

Value

An object of class "unsurv_compare" with elements:

- `summary`: one row per method with K, cluster-size range, and ARI against the reference partition.
- `cluster_summary`: one row per method/cluster with size, event count, and Kaplan-Meier median survival.
- `labels, time, status, reference`: the inputs, stored for plotting.

Examples

```

if (requireNamespace("survival", quietly = TRUE)) {
  set.seed(1)
  n <- 120
  time <- stats::rexp(n, 0.1)
  status <- sample(0:1, n, TRUE)
  labs <- list(
    unsurv_curve = sample(1:3, n, TRUE),
    scalar_risk = sample(1:3, n, TRUE)
  )
  cmp <- unsurv_compare(labs, time, status)
  print(cmp)
}

```

unsurv_stability

Stability assessment for an unsurv clustering

Description

Computes a resampling-based stability score for a fitted unsurv model using the Adjusted Rand Index (ARI) computed on overlap sets across resamples.

Usage

```

unsurv_stability(
  S,
  times,
  fit,
  B = 30,
  frac = 0.5,
  mode = c("bootstrap", "subsample"),
  jitter_sd = 0.01,
  weight_perturb = 0.3,
  eps_jitter = 0.02,
  return_distribution = TRUE
)

```

Arguments

S	Numeric matrix of survival probabilities used for stability assessment ($n \times m$), with columns matching times.
times	Numeric vector of time grid points (length m).
fit	An object of class "unsurv", typically returned by unsurv .
B	Integer; number of resamples.
frac	Numeric in (0, 1]; fraction of rows sampled per resample.

mode	Resampling mode: "bootstrap" (with replacement) or "subsample" (without replacement).
jitter_sd	Nonnegative numeric; curve-space noise level applied before clamping/monotone enforcement.
weight_perturb	Numeric in $[0, 1]$; blends trapezoidal weights with a random simplex to perturb the weighting scheme.
eps_jitter	Nonnegative numeric; feature-space jitter used inside the clustering during re-samples.
return_distribution	Logical; if TRUE, returns the full ARI distribution, else returns only the mean.

Value

If `return_distribution = TRUE`, a list with:

- `mean`: mean ARI across resample-pair overlaps
- `aris`: numeric vector of ARIs

Otherwise, returns a single numeric mean ARI.

Examples

```
if (requireNamespace("cluster", quietly = TRUE)) {
  set.seed(2025)
  n <- 60; Q <- 40
  times <- seq(0, 5, length.out = Q)
  rates <- c(0.12, 0.38, 0.8)
  grp <- sample(1:3, n, TRUE, c(0.4, 0.4, 0.2))
  S <- t(vapply(1:n, function(i)
    pmin(pmax(exp(-rates[grp[i]] * times) + rnorm(Q, 0, 0.01), 0), 1),
    numeric(Q)
  ))

  fit <- unsurv(S, times, K = NULL, K_max = 6, distance = "L2",
    enforce_monotone = TRUE, standardize_cols = FALSE,
    eps_jitter = 0, seed = NULL)

  stab <- unsurv_stability(S, times, fit, B = 8, frac = 0.55, mode = "bootstrap",
    jitter_sd = 0.3, weight_perturb = 0.0, eps_jitter = 0.3,
    return_distribution = TRUE)

  stab$mean
}
```

Index

`autoplot.unsurv`, [3](#)
`autoplot.unsurv_compare`, [4](#)

`matplot`, [5](#)

`plot.unsurv`, [3](#), [5](#)
`plot.unsurv_compare`
 (`autoplot.unsurv_compare`), [4](#)
`plot_stability`, [6](#)
`predict`, [11](#)
`predict.unsurv`, [3](#), [6](#)
`print.summary.unsurv`, [7](#)
`print.unsurv`, [8](#)
`print.unsurv_compare`, [9](#)

`summary.unsurv`, [3](#), [9](#)

`unsurv`, [2](#), [7](#), [10](#), [14](#)
`unsurv-package`, [2](#)
`unsurv_compare`, [4](#), [13](#)
`unsurv_stability`, [3](#), [14](#)