Package 'unsurv'

May 8, 2026

Title Unsupervised Clustering of Individualized Survival Curves

Version 0.5.0

Author Imad EL BADISY [aut, cre]

Maintainer Imad EL BADISY <elbadisyimad@gmail.com>

Date 2026-03-12

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 clustering strategy follows Kaufman and Rousseeuw (1990, ISBN:0471878766).

License MIT + file LICENSE

URL https://github.com/ielbadisy/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 tidyr, dplyr, scales, testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-03-17 18:00:03 UTC

Contents

`unsurv-package` *unsurv: Unsupervised clustering of individualized survival curves*

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://github.com/ielbadisy/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)
}
```

`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 *Plot stability distribution*

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 `unsurv_stability(..., return_distribution = TRUE)`.

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 *Predict cluster membership for new survival curves*

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)
}
```

`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, ..., 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

`S`	Numeric matrix of survival probabilities with shape $n \times m$. Rows are subjects, columns correspond to `times`. Values are clamped to $[0, 1]$.

`times`	Numeric vector of length m (strictly increasing time grid).
`K`	Optional integer number of clusters. If `NULL`, selected by silhouette.
`K_max`	Maximum `K` considered when `K` is `NULL`.
`distance`	Distance type: `"L2"` (euclidean) or `"L1"` (manhattan).
`weights`	Optional nonnegative vector of length m for time-point weights. If `NULL`, trapezoidal weights are used.
`enforce_monotone`	Logical; enforce non-increasing survival curves over time.
`smooth_median_width`	Integer; if ≥ 3 and odd, apply median smoothing along time.
`standardize_cols`	Logical; standardize feature columns before clustering.
`eps_jitter`	Nonnegative numeric; feature-space Gaussian jitter sd to break ties.
`seed`	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, ..., 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_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 resamples.
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