Package: Silhouette (via r-universe)

November 14, 2026

Type Package

Title Proximity Measure Based Diagnostics for Standard, Soft, and Multi-Way Clustering

Version 0.9.7

Depends R (>= 4.2.0)

Language en-US

Maintainer Shrikrishna Bhat K <skbhat.in@gmail.com>

Description Quantifies clustering quality by measuring both cohesion within clusters and separation between clusters. Silhouette widths are computed from the n by K matrices that clustering algorithms return (distances or similarities to cluster prototypes, fuzzy memberships, or posterior probabilities) instead of all pairwise dissimilarities, for the partition an algorithm produced or one supplied by the user. Implements the simplified silhouette of Van der Laan et al. (2003) <doi:10.1080/00949650310001360l2>, the probability of alternative cluster normalization of Raymaekers and Rousseeuw (2022) <doi:10.1080/10618600.2022.2050249>, the fuzzy silhouette average of Campello and Hruschka (2006) <doi:10.1016/j.fss.2006.07.006>, the density-based silhouette of Menardi (2011) <doi:10.1007/s11222-010-9169-0>, the soft silhouettes of Bhat Kapu and Kiruthika (2024) <doi:10.1080/23737484.2024.2408534>, and the extended silhouette for multi-way clustering of Schepers et al. (2008) <doi:10.1007/s00357-008-9005-9> and Bhat Kapu and Kiruthika (2025) <doi:10.21203/rs.3.rs-6973596/v1>. Provides summaries and 'ggplot2' silhouette plots in the style of Rousseeuw (1987) <doi:10.1016/0377-0427(87)90125-7>.

License GPL-2

URL https://kskbhat.github.io/Silhouette/

BugReports https://github.com/kskbhat/Silhouette/issues

Encoding UTF-8

Imports ggplot2, ggpubr, methods, stats, utils

Suggests proxy, ppclust, cluster, factoextra, drclust, mclust, knitr, lifecycle, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website FPDclustering, bioc::bluster

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libicu-dev

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

Date/Publication 2026-11-14 10:31:11 UTC

RemoteUrl https://github.com/kskbhat/silhouette

RemoteRef HEAD

RemoteSha 172e0aea7827388dee69f7d419fb41c678230559

Contents

calSilhouette 2
cerSilhouette 6
dbSilhouette 9
extSilhouette 13
getSilhouette 15
is.Silhouette 17
plotSilhouette 18
Silhouette 21
softSilhouette 28

Index **33**

`calSilhouette` *Calculate All Available Silhouette Summaries*

Description

Computes every silhouette width available in the package for one clustering and returns a data frame comparing the crisp, fuzzy, and median averages of each.

Usage

```
calSilhouette(
  prox_matrix = NULL,
  proximity_type = c("dissimilarity", "similarity"),
  prob_matrix = NULL,
  a = 2,
  print.summary = FALSE,
  clust_fun = NULL,
  ...,
  cluster = NULL,
  clust_args = list()
)
```

Arguments

prox_matrix	A numeric matrix where rows represent observations and columns represent proximity measures (e.g., distances or similarities) to clusters. Typically, this is a membership or dissimilarity matrix from clustering results. If clust_fun is provided, prox_matrix should be the name of the matrix component as a string (e.g., if clust_fun = fcm from **ppclust** package the prox_matrix = "d"). Defaults to NULL.
proximity_type	Character string specifying the type of proximity measure in prox_matrix. Options are "similarity" (higher values indicate closer proximity) or "dissimilarity" (lower values indicate closer proximity). Defaults to "dissimilarity".
prob_matrix	A numeric matrix of cluster membership probabilities, where rows represent observations and columns represent clusters in the same order as prox_matrix. If clust_fun is provided, prob_matrix can be given as the name of the matrix component (e.g., "u" for the fcm function). Defaults to NULL.
a	Non-negative number, the exponent α of the weights in fuzzy silhouette averaging. Higher values increase the emphasis on strong membership differences; a = 0 gives every observation the same weight, so the fuzzy average equals the crisp average. Defaults to 2.
print.summary	Logical; if TRUE, reports the summary table through message. Defaults to FALSE.
clust_fun	Optional S3 or S4 function object or function as character string specifying a clustering function that produces the proximity measure matrix. For example, fcm or "fcm". If provided, prox_matrix must be the name of the matrix component in the clustering output (e.g., "d" for fcm when proximity_type = "dissimilarity"). Defaults to NULL.
...	Additional arguments passed to clust_fun, such as x,centers for fcm.
cluster	Optional integer vector of length n giving the cluster of each observation as a column number $(1, \ldots, K)$. If clust_fun is provided, cluster may be the name of the label component of its output. Must be named in the call because it follows .... See *Cluster labels*.

`clust_args` A list of further arguments for `clust_fun`. Arguments given through `...` are matched first against this function's own arguments, and R matches names partially: `m = 3` meant for the fuzzifier of `fcm` would be taken as `method`. Pass such arguments here instead, e.g. `clust_args = list(m = 3)`. Defaults to `list()`.

Details

The rows are not independent pieces of evidence: several are exact or monotone transformations of one another (see *Relations between the rows*). The distance-based rows compare distances to the cluster representatives; the probability-based rows are the silhouettes of Bhat Kapu and Kiruthika (2024), which treat the posterior probabilities or the component densities as proximities, and the density-based silhouette of Menardi (2011).

The methods included depend on the available input matrices:

If `prox_matrix` is available:

- `medoid` - Medoid-based silhouette using `Silhouette`
- `pac` - PAC-based silhouette using `Silhouette`

If `prob_matrix` is available:

- `pp_pac` - PACS: posterior probabilities with the PAC normalization (`softSilhouette`)
- `pp_medoid` - PPS: posterior probability silhouette (`softSilhouette`)
- `nlpp_pac` - Negative log posterior probabilities with the PAC normalization (`softSilhouette`)
- `nlpp_medoid` - NLPPS: negative log of posterior probability silhouette (`softSilhouette`)
- `pd_pac` - Probability densities with the PAC normalization (`softSilhouette`)
- `pd_medoid` - PDS: probability density silhouette (`softSilhouette`)
- `cer` - CeS: certainty silhouette (`cerSilhouette`)
- `db` - DBS: density-based silhouette of Menardi (2011) (`dbSilhouette`)

At least one of `prox_matrix` or `prob_matrix` must be provided. Each width is computed once and summarized by all available averages; the fuzzy average needs `prob_matrix`.

If a width cannot be computed, its row is `NA` and a warning gives the reason. Warnings from the individual functions are collected and each distinct warning is reported once.

Value

A data frame with one row per method and the following columns:

Method Character vector of method names, as listed in Details.

Crisp Crisp (unweighted mean) average silhouette width.

Fuzzy Fuzzy (membership-weighted) average silhouette width; present only when `prob_matrix` is supplied.

Median Median silhouette width.

Cluster labels

When `cluster` is supplied, every row scores that partition. Otherwise every row except the PDS rows (`pd_pac`, `pd_medoid`) uses the largest membership in `prob_matrix`, or the closest cluster in `prox_matrix` when there are no memberships. The PDS rows assign each observation to its largest density, as in the definition of the PDS (see `softSilhouette`); this agrees with the largest membership when the clusters have equal estimated weights. (In version 0.9.6 and earlier, the crisp and median distance-based rows used the closest cluster in `prox_matrix` even when `prob_matrix` was supplied.)

Relations between the rows

For the same labels, the rows fall into the following groups.

Monotone transformations of each other (same ranking of observations). Each `pac` width is $s/(2-|s|)$ of the corresponding `medoid` width s: `medoid`/`pac`, `pp_medoid`/`pp_pac`, `nlpp_medoid`/`nlpp_pac` and `pd_medoid`/`pd_pac`. With the default labels, `pp_medoid`, `pp_pac` and `db` are all increasing functions of the ratio of the two largest memberships, so they too rank observations identically. The averages still differ, and may peak at different numbers of clusters.

Equal in special cases. `pd` (PDS) equals `pp` (PPS) when all estimated cluster weights are equal. For fuzzy c-means with fuzzifier m, `pp_medoid` and `pp_pac` equal `medoid` and `pac` computed from the dissimilarities $d_{ik}^{1/(m-1)}$, where d_{ik} are squared Euclidean distances to the prototypes (for $m = 2$, the `d` component of `fcm`).

Genuinely different. The distance-based rows (in general), `nlpp` (which also depends on the size of the neighboring membership), and `cer` (the largest membership alone, bounded below by $1/K$).

Choosing the number of clusters

Computing `calSilhouette()` over $k = 2, 3, \ldots$ gives one curve per row. The distance-based rows are usually read at their maximum. Bhat Kapu and Kiruthika (2024) read the probability-based rows from the reverse elbow drop of the curve, and the DBS from its local maxima (Menardi, 2011). The suggested number of clusters depends on the algorithm that produced the memberships, on the width and on the averaging.

References

Bhat Kapu, S., & Kiruthika. (2024). Some density-based silhouette diagnostics for soft clustering algorithms. Communications in Statistics: Case Studies, Data Analysis and Applications, 10(3-4), 221-238. doi:10.1080/23737484.2024.2408534

Menardi, G. (2011). Density-based silhouette diagnostics for clustering methods. *Statistics and Computing*, 21(3), 295–308. doi:10.1007/s1122201091690

See Also

`Silhouette`, `softSilhouette`, `dbSilhouette`, `cerSilhouette`

Examples

```
if (requireNamespace("ppclust", quietly = TRUE)) {
  # Example with FCM clustering
  library(ppclust)
  data(iris)
  fcm_result <- fcm(iris[, -5], centers = 3)

  # Using matrices directly
  summary_result <- calSilhouette(
    prox_matrix = fcm_result$d,
    prob_matrix = fcm_result$u,
    proximity_type = "dissimilarity",
    print.summary = TRUE
  )
}

if (requireNamespace("ppclust", quietly = TRUE)) {
  # Using clustering function
  summary_result2 <- calSilhouette(
    prox_matrix = "d",
    prob_matrix = "u",
    proximity_type = "dissimilarity",
    clust_fun = ppclust::fcm,
    x = iris[, -5],
    centers = 3,
    print.summary = TRUE
  )
}
```

---

cerSilhouette *Certainty Silhouette Width (CeS) for Soft Clustering*

Description

Computes the certainty silhouette (CeS) of Bhat Kapu and Kiruthika (2024) for each observation: the membership probability of its assigned cluster (by default, its largest membership).

Usage

```
cerSilhouette(
  prob_matrix,
  average = c("crisp", "fuzzy", "median"),
  a = 2,
  sort = FALSE,
  print.summary = FALSE,
  clust_fun = NULL,
  ...,
  cluster = NULL,
  clust_args = list()
)
```

Arguments

prob_matrix	A numeric matrix of posterior probabilities where rows represent observations and columns represent clusters. Must sum to 1 by row. If clust_fun is provided, prob_matrix must be a string giving the name of the matrix component (e.g., "u").
average	Character string specifying the method for computing the average silhouette width. Options are: • "crisp" – unweighted (simple) average. • "fuzzy" – weighted average; the weight of an observation is the gap between its two largest memberships raised to the power a. • "median" – median silhouette width across observations. Defaults to "crisp".
a	Non-negative number, the exponent α of the weights in fuzzy silhouette averaging. Higher values increase the emphasis on strong membership differences; a = 0 gives every observation the same weight, so the fuzzy average equals the crisp average. Defaults to 2.
sort	Logical; if TRUE, sorts the output widths data frame by cluster and descending silhouette width. Defaults to FALSE.
print.summary	Logical; if TRUE, reports a summary table of average silhouette widths and sizes for each cluster through message. Defaults to FALSE for Silhouette() and TRUE for summary().
clust_fun	Optional S3 or S4 function object or function as character string specifying a clustering function that produces the membership matrix. For example, fcm or "fcm". If provided, prob_matrix must be a string giving the name of the matrix component (e.g., "u"). Defaults to NULL.
...	Additional arguments passed to clust_fun, such as x, centers for fcm.
cluster	Optional integer vector of length n giving the cluster of each observation as a column number of prob_matrix. If NULL (the default), each observation is assigned to the cluster with its largest membership (for "pp" and "nlpp") or its largest density (for "pd"); see *Cluster labels*. If clust_fun is provided, cluster may be the name of the label component of its output. Must be named in the call because it follows ....
clust_args	A list of further arguments for clust_fun; use it for arguments whose names could be partially matched to this function's own (for example clust_args = list(m = 3) for the fuzzifier of fcm). Defaults to list().

Details

The certainty silhouette is the counterpart of the uncertainty used in model-based clustering: values near 1 indicate high certainty that an observation belongs to its cluster, and values near $1/K$ ambiguity. It is bounded below by $1/K$, so its level falls as K grows; Bhat Kapu and Kiruthika (2024) read the number of clusters from the reverse elbow drop of its average over k (see softSilhouette).

Let the posterior probability matrix or cluster membership matrix be $\Gamma = [\gamma_{ik}]_{n\times K}$. The **certainty silhouette width** for observation i assigned to cluster c_i is:

$$\mathrm{Cer}_i = \gamma_{ic_i},$$

which equals $\max_k \gamma_{ik} \in [1/K, 1]$ with the default labels. With labels supplied through `cluster`, the width lies in $[0, 1]$. The `neighbor` column gives the cluster with the largest membership other than c_i.

Ties are resolved in favor of the first column; see `Silhouette`.

Value

A data frame of class `"Silhouette"` with one row per observation and columns `cluster`, `neighbor`, `weight` (only when `average = "fuzzy"`) and `sil_width`. The object includes the following attributes:

proximity_type The proximity type used i.e., `"similarity"`.

method The silhouette calculation method used i.e., `"certainty"`.

average Character — the averaging method: `"crisp"`, `"fuzzy"`, or `"median"`.

Averages

If `average = "crisp"`, the **crisp silhouette index** (CS) is the mean width:

$$CS = \frac{1}{n} \sum_{i=1}^{n} S(x_i)$$

summarizing overall clustering quality.

If `average = "fuzzy"` and `prob_matrix` is provided, denoted as $\Gamma = [\gamma_{ik}]_{n \times K}$, with γ_{ik} representing the probability of observation i belonging to cluster k, the **fuzzy silhouette index** (FS) of Campello and Hruschka (2006) is:

$$FS = \frac{\sum_{i=1}^{n} w_i S(x_i)}{\sum_{i=1}^{n} w_i}, \qquad w_i = (\gamma_{i(1)} - \gamma_{i(2)})^{\alpha},$$

where $\gamma_{i(1)}$ and $\gamma_{i(2)}$ are the largest and second largest memberships of observation i, and α (the `a` argument) controls the emphasis on confident assignments ($\alpha \geq 0$; $\alpha = 0$ gives the crisp average). The weight w_i is stored in the `weight` column and does not depend on the cluster labels; it equals Equation (8) of Bhat Kapu and Kiruthika (2024) when each observation is assigned to its largest membership. If all weights are zero, the fuzzy average is undefined (`NaN`) and a warning is given.

If `average = "median"`, the median width is reported.

Missing widths are removed before averaging.

References

Bhat Kapu, S., & Kiruthika. (2024). Some density-based silhouette diagnostics for soft clustering algorithms. Communications in Statistics: Case Studies, Data Analysis and Applications, 10(3-4), 221-238. doi:10.1080/23737484.2024.2408534

See Also

`Silhouette`, `softSilhouette`, `dbSilhouette`, `calSilhouette`, `getSilhouette`, `is.Silhouette`, `plotSilhouette`

Examples

```
prob <- matrix(c(0.90, 0.10,
                 0.60, 0.40,
                 0.20, 0.80,
                 0.45, 0.55), ncol = 2, byrow = TRUE)
cerSilhouette(prob)


# Compare two soft clustering algorithms on the iris data
data(iris)
if (requireNamespace("ppclust", quietly = TRUE)) {
  fcm_result <- ppclust::fcm(iris[, 1:4], 3)
  out_fcm <- cerSilhouette(prob_matrix = fcm_result$u, print.summary = TRUE)
  plot(out_fcm)
  fcm2_result <- ppclust::fcm2(iris[, 1:4], 3)
  out_fcm2 <- cerSilhouette(prob_matrix = fcm2_result$u)
  c(fcm = summary(out_fcm, print.summary = FALSE)$avg.width,
    fcm2 = summary(out_fcm2, print.summary = FALSE)$avg.width)
} else {
  message("Install 'ppclust' to run this example: install.packages('ppclust')")
}
```

`dbSilhouette` *Density-Based Silhouette Width (DBS) for Soft Clustering*

Description

Computes the density-based silhouette (DBS) of Menardi (2011), using log-ratios of posterior probabilities.

Usage

```
dbSilhouette(
  prob_matrix,
  average = c("median", "crisp", "fuzzy"),
  a = 2,
  sort = FALSE,
  print.summary = FALSE,
  clust_fun = NULL,
  ...,
  cluster = NULL,
  clust_args = list()
)
```

Arguments

prob_matrix	A numeric matrix of posterior probabilities where rows represent observations and columns represent clusters. Must sum to 1 by row. If clust_fun is provided, prob_matrix must be a string giving the name of the matrix component (e.g., "u").
average	Character string specifying the method for computing the average silhouette width. Options are: • "crisp" – unweighted (simple) average. • "fuzzy" – weighted average; the weight of an observation is the gap between its two largest memberships raised to the power a. • "median" – median silhouette width across observations. Defaults to "median".
a	Non-negative number, the exponent α of the weights in fuzzy silhouette averaging. Higher values increase the emphasis on strong membership differences; a = 0 gives every observation the same weight, so the fuzzy average equals the crisp average. Defaults to 2.
sort	Logical; if TRUE, sorts the output widths data frame by cluster and descending silhouette width. Defaults to FALSE.
print.summary	Logical; if TRUE, reports a summary table of average silhouette widths and sizes for each cluster through message. Defaults to FALSE for Silhouette() and TRUE for summary().
clust_fun	Optional S3 or S4 function object or function as character string specifying a clustering function that produces the membership matrix. For example, fcm or "fcm". If provided, prob_matrix must be a string giving the name of the matrix component (e.g., "u"). Defaults to NULL.
...	Additional arguments passed to clust_fun, such as x, centers for fcm.
cluster	Optional integer vector of length n giving the cluster of each observation as a column number of prob_matrix. If NULL (the default), each observation is assigned to the cluster with its largest membership (for "pp" and "nlpp") or its largest density (for "pd"); see *Cluster labels*. If clust_fun is provided, cluster may be the name of the label component of its output. Must be named in the call because it follows ....
clust_args	A list of further arguments for clust_fun; use it for arguments whose names could be partially matched to this function's own (for example clust_args = list(m = 3) for the fuzzifier of fcm). Defaults to list().

Details

The DBS compares the largest membership with the second largest on the log scale, scaled by the largest such log ratio in the sample. Its numerator is that of the NLPPS of Bhat Kapu and Kiruthika (2024), with a different denominator. Following Menardi (2011), the median is the default summary, and the number of clusters is read from the local maxima of the average over k.

Let the posterior probability matrix or cluster membership matrix be $\Gamma = [\gamma_{ik}]_{n\times K}$, let c_i be the cluster of observation i and ν_i the cluster with its largest membership other than c_i. With the log

ratio

$$R_i = \log \frac{\gamma_{ic_i}}{\gamma_{i\nu_i}},$$

the **density-based silhouette width** is

$$\mathrm{DBS}_i = \frac{R_i}{\max_{j=1,\ldots,n} |R_j|}.$$

With the default labels (largest membership) $R_i \geq 0$, so the widths lie in $[0, 1]$ and the most confidently assigned observation has width 1. With labels supplied through `cluster`, R_i can be negative and the widths lie in $[-1, 1]$.

Because of the scaling, the average width depends on how extreme the most confident observation is: a single observation with a very large log ratio shrinks every other width towards zero. Compare the average only across fits with similar extremes, and report it together with the mode-wise or cluster-wise summaries.

Value

A data frame of class `"Silhouette"` with one row per observation and columns `cluster`, `neighbor`, `weight` (only when `average = "fuzzy"`) and `sil_width`. The object includes the following attributes:

proximity_type The proximity type used i.e., `"similarity"`.

method The silhouette calculation method used i.e., `"db"`.

average Character — the averaging method: `"crisp"`, `"fuzzy"`, or `"median"`.

Numerical safeguards

The following rules keep the widths finite and meaningful:

- **Zero probabilities.** When the neighboring membership is exactly zero (for example, after a posterior probability underflows), $R_i = +\infty$. Such observations receive width 1 (and width -1 when the assigned membership is zero, which can happen only with supplied labels), and a warning reports how many there are. The remaining widths are scaled by the largest *finite* $|R_j|$. (In version 0.9.6 and earlier, the infinite maximum turned every finite width into 0.)
- **Equal memberships.** When the two memberships are equal, $R_i = 0$ and the width is 0.
- **Degenerate fits.** When the largest finite $|R_j|$ is below `sqrt(.Machine$double.eps)` (about 1.5×10^{-8}), the memberships are numerically indistinguishable and scaling would only magnify rounding error. The finite widths are then set to 0 with a warning.

Ties are resolved in favor of the first column; see `Silhouette`.

Averages

If `average = "crisp"`, the **crisp silhouette index** (CS) is the mean width:

$$CS = \frac{1}{n} \sum_{i=1}^{n} S(x_i)$$

summarizing overall clustering quality.

If `average = "fuzzy"` and `prob_matrix` is provided, denoted as $\Gamma = [\gamma_{ik}]_{n \times K}$, with γ_{ik} representing the probability of observation i belonging to cluster k, the **fuzzy silhouette index** (FS) of Campello and Hruschka (2006) is:

$$FS = \frac{\sum_{i=1}^{n} w_i S(x_i)}{\sum_{i=1}^{n} w_i}, \qquad w_i = (\gamma_{i(1)} - \gamma_{i(2)})^{\alpha},$$

where $\gamma_{i(1)}$ and $\gamma_{i(2)}$ are the largest and second largest memberships of observation i, and α (the `a` argument) controls the emphasis on confident assignments ($\alpha \geq 0$; $\alpha = 0$ gives the crisp average). The weight w_i is stored in the `weight` column and does not depend on the cluster labels; it equals Equation (8) of Bhat Kapu and Kiruthika (2024) when each observation is assigned to its largest membership. If all weights are zero, the fuzzy average is undefined (`NaN`) and a warning is given.

If `average = "median"`, the median width is reported.

Missing widths are removed before averaging.

References

Menardi, G. (2011). Density-based silhouette diagnostics for clustering methods. *Statistics and Computing*, 21(3), 295–308. doi:10.1007/s1122201091690

Bhat Kapu, S., & Kiruthika. (2024). Some density-based silhouette diagnostics for soft clustering algorithms. Communications in Statistics: Case Studies, Data Analysis and Applications, 10(3-4), 221-238. doi:10.1080/23737484.2024.2408534

See Also

`Silhouette`, `softSilhouette`, `cerSilhouette`, `calSilhouette`, `getSilhouette`, `is.Silhouette`, `plotSilhouette`

Examples

```
prob <- matrix(c(0.90, 0.10,
                 0.60, 0.40,
                 0.20, 0.80,
                 0.45, 0.55), ncol = 2, byrow = TRUE)
dbSilhouette(prob)

# A zero neighboring probability gives width 1 and a warning
prob0 <- rbind(prob, c(1, 0))
dbSilhouette(prob0)


# Compare two soft clustering algorithms on the iris data
data(iris)
if (requireNamespace("ppclust", quietly = TRUE)) {
  fcm_result <- ppclust::fcm(iris[, 1:4], 3)
  out_fcm <- dbSilhouette(prob_matrix = fcm_result$u, print.summary = TRUE)
  plot(out_fcm)
  fcm2_result <- ppclust::fcm2(iris[, 1:4], 3)
  out_fcm2 <- dbSilhouette(prob_matrix = fcm2_result$u)
  c(fcm = summary(out_fcm, print.summary = FALSE)$avg.width,
```

```
    fcm2 = summary(out_fcm2, print.summary = FALSE)$avg.width)
} else {
  message("Install 'ppclust' to run this example: install.packages('ppclust')")
}
```

`extSilhouette` *Calculate Extended Silhouette Width for Multi-Way Clustering*

Description

[Experimental]

Computes an extended silhouette width for multi-way clustering (e.g., biclustering, triclustering, or n-mode tensor clustering) by combining silhouette widths from a list of Silhouette objects, each representing one mode of clustering. The extended silhouette width is a weighted average of the average silhouette widths of the modes. By default each mode is weighted by its number of objects; equal or user-specified weights are also available. The output is an object of class `extSilhouette`.

Usage

```
extSilhouette(
  sil_list,
  dim_names = NULL,
  print.summary = FALSE,
  weights = c("size", "equal")
)

## S3 method for class 'extSilhouette'
format(x, digits = 4, ...)

## S3 method for class 'extSilhouette'
print(x, digits = 4, ...)
```

Arguments

`sil_list`	A list of objects of class `"Silhouette"`, typically the output of `Silhouette` or `softSilhouette`, where each object represents the silhouette analysis for one mode of multi-way clustering (e.g., rows, columns, or other dimensions in biclustering or tensor clustering).
`dim_names`	An optional character vector of dimension names (e.g., `c("Rows", "Columns")`). If `NULL`, defaults to `"Mode 1"`, `"Mode 2"`, etc.
`print.summary`	Logical; if `TRUE`, reports the extended silhouette width and the dimension table through `message`. Default is `FALSE`.
`weights`	How the modes are weighted: `"size"` (the default) weights each mode by its number of objects, `"equal"` gives every mode the same weight, and a non-negative numeric vector with one entry per mode gives user-specified weights (they are rescaled to sum to 1).

x	An object of class "extSilhouette".
digits	Number of decimal places shown.
...	Further arguments, currently ignored.

Details

The extended silhouette width is computed as:

$$ExS = \frac{\sum_i \omega_i \bar{S}_i}{\sum_i \omega_i}$$

where $\bar{S}_i$ is the average silhouette width of mode i (from `summary(x)$avg.width`, so it follows the averaging rule of that object) and ω_i is its weight. With `weights = "size"`, $\omega_i = n_i$, the number of objects in mode i (`nrow(x)`), as in Schepers et al. (2008); ExS is then the average width over all objects of all modes, each object counting once. With `weights = "equal"`, $\omega_i = 1$, so ExS is the mean of the mode-wise averages.

Size weighting treats every clustered object alike, but when the modes differ greatly in size the largest mode dominates: with 200 rows and 30 columns, the rows receive about 87\ of the weight. Equal weighting gives every mode the same say regardless of its size. Because the choice can change which model is selected, report the mode-wise averages in `dim_table` together with ExS.

Each `Silhouette` object in `sil_list` must contain at least one observation and have a finite average width. For consistency, compute all `Silhouette` objects with the same function, `method` and averaging rule.

Value

A list of class "extSilhouette" with the following components:

ext_sil_width A numeric scalar, the extended silhouette width.

dim_table A data frame with columns `dimension` (e.g., "Mode 1", "Mode 2"), `n_obs` (number of objects), `weight` (the share of the total weight given to the mode) and `avg_sil_width` (average silhouette width of the mode).

weights The weighting used: "size", "equal" or "user".

Values are not rounded; the print method shows four decimals.

References

Schepers, J., Ceulemans, E., & Van Mechelen, I. (2008). Selecting among multi-mode partitioning models of different complexities: A comparison of four model selection criteria. *Journal of Classification*, 25(1), 67–85. doi:10.1007/s0035700890059

Bhat Kapu, S., & Kiruthika, C. (2025). Block Probabilistic Distance Clustering: A Unified Framework and Evaluation. PREPRINT (Version 1) available at Research Square. doi:10.21203/rs.3.rs-6973596/v1

See Also

`Silhouette`, `softSilhouette`, `dbSilhouette`, `cerSilhouette`, `getSilhouette`, `is.Silhouette`

Examples

```
# Two modes: 150 objects in 3 clusters and 4 variables in 2 clusters
set.seed(123)
prob_mode1 <- matrix(runif(150 * 3), nrow = 150, ncol = 3)
prob_mode1 <- prob_mode1 / rowSums(prob_mode1)
prob_mode2 <- matrix(runif(4 * 2), nrow = 4, ncol = 2)
prob_mode2 <- prob_mode2 / rowSums(prob_mode2)

sil_mode1 <- softSilhouette(prob_matrix = prob_mode1, method = "pac")
sil_mode2 <- softSilhouette(prob_matrix = prob_mode2, method = "pac")

# Size-weighted (default) and equally weighted extended silhouette
extSilhouette(list(sil_mode1, sil_mode2), dim_names = c("Rows", "Columns"))
extSilhouette(list(sil_mode1, sil_mode2), dim_names = c("Rows", "Columns"),
              weights = "equal")
```

---

getSilhouette *Create Silhouette Object from User Components*

Description

Constructs a Silhouette class object directly from user-provided components without performing silhouette calculations. This function allows users to build a Silhouette object when they already have the necessary components.

Usage

```
getSilhouette(
  cluster,
  neighbor,
  sil_width,
  weight = NULL,
  proximity_type = c("dissimilarity", "similarity"),
  method = NA,
  average = c("crisp", "fuzzy", "median")
)
```

Arguments

`cluster`	Numeric or integer vector of cluster assignments for each observation
`neighbor`	Numeric or integer vector of nearest neighbor cluster assignments for each observation
`sil_width`	Numeric vector of silhouette widths for each observation (must be between -1 and +1)
`weight`	Numeric vector of weights for each observation (must be between 0 and 1, only used when average = "fuzzy")
`proximity_type`	Character; the proximity type used. Options: "similarity" or "dissimilarity"

method	Character; the silhouette calculation method used (default: NA; any custom name is allowed)
average	Character; the averaging method. Options: "crisp", "fuzzy", or "median"

Value

A data frame of class "Silhouette" containing cluster assignments, nearest neighbor clusters, silhouette widths for each observation, and weights (for fuzzy clustering). The object includes the following attributes:

proximity_type The proximity type used ("similarity" or "dissimilarity").

method The silhouette calculation method supplied in method.

average Character — the averaging method: "crisp", "fuzzy", or "median".

See Also

Silhouette, softSilhouette, dbSilhouette, cerSilhouette, is.Silhouette, plotSilhouette

Examples

```
# Create a simple crisp Silhouette object (3 columns)
cluster_assignments <- c(1, 1, 2, 2, 3, 3)
neighbor_clusters <- c(2, 2, 1, 1, 1, 1)
silhouette_widths <- c(0.8, 0.7, 0.6, 0.9, 0.5, 0.4)

sil_obj <- getSilhouette(
  cluster = cluster_assignments,
  neighbor = neighbor_clusters,
  sil_width = silhouette_widths,
  proximity_type = "dissimilarity",
  method = "medoid",
  average = "crisp"
)
sil_obj

# Create a fuzzy Silhouette object with weights (4 columns)
weights <- c(0.9, 0.8, 0.7, 0.95, 0.6, 0.5)

sil_fuzzy <- getSilhouette(
  cluster = cluster_assignments,
  neighbor = neighbor_clusters,
  sil_width = silhouette_widths,
  weight = weights,
  proximity_type = "similarity",
  method = "pac",
  average = "fuzzy"
)
sil_fuzzy

# Custom method name
sil_custom <- getSilhouette(
```

```
  cluster = cluster_assignments,
  neighbor = neighbor_clusters,
  sil_width = silhouette_widths,
  proximity_type = "dissimilarity",
  method = "my_custom_method",
  average = "crisp"
)
sil_custom
```

`is.Silhouette` *Check if Object is of Class Silhouette*

Description

Tests whether an object is of class "Silhouette". This function checks both the class inheritance and the expected structure of a Silhouette object.

Usage

```
is.Silhouette(x, strict = FALSE)
```

Arguments

`x`	An object to test
`strict`	Logical; if TRUE, performs additional structural validation beyond just class checking (default: FALSE)

Details

When `strict = FALSE`, the function only checks if the object inherits from the "Silhouette" class.

When `strict = TRUE`, the function additionally validates:

- Object is a data frame
- Has required columns: cluster, neighbor, sil_width
- Has required attributes: proximity_type, method, average
- Column types are appropriate (integer for cluster/neighbor, numeric for sil_width)

The Silhouette object attributes are validated as follows:

- `proximity_type`: Must be one of "dissimilarity" or "similarity"
- `average`: Must be one of "crisp", "fuzzy", or "median"
- `method`: any string, or `NA`

Value

Logical; TRUE if the object is of class "Silhouette", FALSE otherwise

See Also

Silhouette, softSilhouette, dbSilhouette, cerSilhouette, getSilhouette, plotSilhouette

Examples

```
# Create a Silhouette object
cluster_assignments <- c(1, 1, 2, 2, 3, 3)
neighbor_clusters <- c(2, 2, 1, 1, 1, 1)
silhouette_widths <- c(0.8, 0.7, 0.6, 0.9, 0.5, 0.4)

sil_obj <- getSilhouette(
  cluster = cluster_assignments,
  neighbor = neighbor_clusters,
  sil_width = silhouette_widths,
  proximity_type = "dissimilarity",
  method = "medoid",
  average = "crisp"
)

# Test if object is Silhouette
is.Silhouette(sil_obj)          # TRUE
is.Silhouette(sil_obj, strict = TRUE)  # TRUE

# Test with non-Silhouette objects
is.Silhouette(data.frame(a = 1, b = 2))  # FALSE
is.Silhouette(matrix(1:10, ncol = 2))    # FALSE
is.Silhouette(list(a = 1, b = 2))        # FALSE
is.Silhouette(NULL)                       # FALSE
```

---

plotSilhouette *Plot Silhouette Analysis Results*

Description

Creates a silhouette plot for visualizing the silhouette widths of clustering results, with bars colored by cluster and an optional summary of cluster statistics in legend.

Usage

```
plotSilhouette(
  x,
  label = FALSE,
  summary.legend = TRUE,
  grayscale = FALSE,
  linetype = c("dashed", "solid", "dotted", "dotdash", "longdash", "twodash"),
  ...
)
```

Arguments

x	An object of class "Silhouette", typically the output of the Silhouette, softSilhouette, dbSilhouette and cerSilhouette function. Also supports objects classes eclust, hcut, pam, clara, fanny, silhouette, or silhouette from **cluster**, **factoextra**, **drclust** packages. For these classes, explicitly call plotSilhouette() to generate the plot.
label	Logical; if TRUE, the x-axis is labeled with observation row indices from the input data and titled "Row Index". Defaults to FALSE.
summary.legend	Logical; if TRUE, prints a summary of average silhouette widths and sizes for each cluster in legend ("Cluster (Size): Width"). If FALSE, the legend shows only cluster numbers. Defaults to TRUE.
grayscale	Logical; if TRUE, the plot uses a grayscale color palette for clusters. If FALSE, uses the default or specified color palette. Defaults to FALSE.
linetype	Character or numeric value specifying the type of line to be used for the horizontal reference line indicating the average silhouette width. Accepts standard ggplot2 linetype values, such as: • Character: "solid", "dashed", "dotted", "dotdash", "longdash", "twodash". • Numeric: a single integer from 0 to 6 corresponding to ggplot2 line patterns. Defaults to "dashed".
...	Additional arguments passed to ggpar for customizing the plot (e.g., legend, xlab, ylab, subtitle, title). A palette (a vector of colors or a palette name accepted by get_palette) colors the clusters and keeps the summary legend; it is ignored when grayscale = TRUE.

Details

The Silhouette plot displays the silhouette width (sil_width) for each observation, grouped by cluster, with bars sorted by cluster and descending silhouette width. The summary.legend option adds cluster sizes and average silhouette widths to the legend.

This function is the S3 plot method for objects of class "Silhouette", typically produced by the Silhouette, softSilhouette, dbSilhouette or cerSilhouette functions in this package. It also supports objects of the following classes, with silhouette information extracted from their respective component:

- "eclust": Produced by eclust from the **factoextra** package.
- "hcut": Produced by hcut from the **factoextra** package.
- "pam": Produced by pam from the **cluster** package.
- "clara": Produced by clara from the **cluster** package.
- "fanny": Produced by fanny from the **cluster** package.
- "silhouette": Produced by silhouette from the **cluster** package or silhouette from the **drclust** package.

For these classes (`"eclust"`, `"hcut"`, `"pam"`, `"clara"`, `"fanny"`, `"silhouette"`), users should explicitly call `plotSilhouette()` (e.g., `plotSilhouette(pam_result)`) to ensure the correct method is used, as the generic `plot()` may not dispatch to this function for these objects.

The design of the plot follows `fviz_silhouette` in the **factoextra** package (Kassambara and Mundt, 2020), extended to the `"Silhouette"` class, fuzzy and median averages, and grayscale output.

Value

A `ggplot2` object representing the Silhouette plot.

References

Kassambara, A., & Mundt, F. (2020). factoextra: Extract and Visualize the Results of Multivariate Data Analyses. R package version 1.0.7. https://CRAN.R-project.org/package=factoextra

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65. doi:10.1016/0377-0427(87)901257

See Also

`Silhouette`, `softSilhouette`, `dbSilhouette`, `cerSilhouette`, `getSilhouette`, `is.Silhouette`

Examples

```
data(iris)

# Crisp Silhouette with k-means
out <- kmeans(iris[, -5], 3)
if (requireNamespace("proxy", quietly = TRUE)) {
  d_km <- proxy::dist(iris[, -5], out$centers)
  plot(Silhouette(d_km))
  plot(Silhouette(d_km), palette = "jco")
}

# Fuzzy Silhouette with ppclust::fcm
if (requireNamespace("ppclust", quietly = TRUE)) {
  library(ppclust)
  out_fuzzy <- Silhouette(
    prox_matrix = "d",
    proximity_type = "dissimilarity",
    prob_matrix = "u",
    clust_fun = ppclust::fcm,
    x = iris[, 1:4],
    centers = 3,
    sort = TRUE
  )
  plot(out_fuzzy, summary.legend = FALSE, grayscale = TRUE)
} else {
  message("Install 'ppclust': install.packages('ppclust')")
}
```

```
# Silhouette plot for pam clustering
if (requireNamespace("cluster", quietly = TRUE)) {
  library(cluster)
  pam_result <- pam(iris[, 1:4], k = 3)
  plotSilhouette(pam_result)
}

# Silhouette plot for clara clustering
if (requireNamespace("cluster", quietly = TRUE)) {
  clara_result <- clara(iris[, 1:4], k = 3)
  plotSilhouette(clara_result)
}

# Silhouette plot for fanny clustering
if (requireNamespace("cluster", quietly = TRUE)) {
  fanny_result <- fanny(iris[, 1:4], k = 3)
  plotSilhouette(fanny_result)
}

# Example using base silhouette() object
if (requireNamespace("cluster", quietly = TRUE)) {
  sil <- silhouette(pam_result)
  plotSilhouette(sil)
}

# Silhouette plot for eclust clustering
if (requireNamespace("factoextra", quietly = TRUE)) {
  library(factoextra)
  eclust_result <- eclust(iris[, 1:4], "kmeans", k = 3, graph = FALSE)
  plotSilhouette(eclust_result)
}

# Silhouette plot for hcut clustering
if (requireNamespace("factoextra", quietly = TRUE)) {
  hcut_result <- hcut(iris[, 1:4], k = 3)
  plotSilhouette(hcut_result)
}

# Silhouette plot for a drclust partition (3 clusters, 2 components)
if (requireNamespace("drclust", quietly = TRUE)) {
  library(drclust)
  iris_mat <- as.matrix(iris[,-5])
  drclust_out <- dpcakm(iris_mat, 3, 2)
  d <- silhouette(iris_mat, drclust_out)
  plotSilhouette(d$cl.silhouette)
}
```

Silhouette *Calculate Silhouette Widths, Summary, and Plot for Clustering Results*

Description

Computes the simplified silhouette width of each observation from an $n \times K$ matrix of proximities between observations and clusters (for example, distances to cluster centers). The width compares the proximity of an observation to its own cluster with its proximity to the closest other cluster. Widths lie in $[-1, 1]$; values near 1 indicate a clear assignment and negative values indicate that another cluster is closer.

Usage

```
Silhouette(
  prox_matrix,
  proximity_type = c("dissimilarity", "similarity"),
  method = c("medoid", "pac"),
  average = c("crisp", "fuzzy", "median"),
  prob_matrix = NULL,
  a = 2,
  sort = FALSE,
  print.summary = FALSE,
  clust_fun = NULL,
  ...,
  cluster = NULL,
  clust_args = list()
)

## S3 method for class 'Silhouette'
plot(
  x,
  label = FALSE,
  summary.legend = TRUE,
  grayscale = FALSE,
  linetype = c("dashed", "solid", "dotted", "dotdash", "longdash", "twodash"),
  ...
)

## S3 method for class 'Silhouette'
summary(object, print.summary = TRUE, ...)
```

Arguments

prox_matrix	A numeric matrix where rows represent observations and columns represent proximity measures (e.g., distances or similarities) to clusters. Typically, this is a membership or dissimilarity matrix from clustering results. If clust_fun is provided, prox_matrix should be the name of the matrix component as a string (e.g., if clust_fun = fcm from **ppclust** package the prox_matrix = "d").
proximity_type	Character string specifying the type of proximity measure in prox_matrix. Options are "similarity" (higher values indicate closer proximity) or "dissimilarity" (lower values indicate closer proximity). Defaults to "dissimilarity".

`method` Character string specifying the normalization of the silhouette width. Options are `"medoid"` (divide by the larger of the two proximities) or `"pac"` (Probability of Alternative Cluster; divide by their sum). See Details. Defaults to `"medoid"`.

`average` Character string specifying the method for computing the average silhouette width. Options are:

- `"crisp"` – unweighted (simple) average.
- `"fuzzy"` – weighted average; the weight of an observation is the gap between its two largest memberships raised to the power `a`.
- `"median"` – median silhouette width across observations.

Defaults to `"crisp"`.

`prob_matrix` A numeric matrix of cluster membership probabilities, where rows represent observations and columns represent clusters in the same order as `prox_matrix`. If `clust_fun` is provided, `prob_matrix` can be given as the name of the matrix component (e.g., `"u"` for the `fcm` function). When given, and `cluster` is not, each observation is assigned to the cluster with its largest membership, and the weights for fuzzy averaging are computed. When `NULL` and `average = "fuzzy"`, the function falls back to computing the `"crisp"` silhouette with a warning, since fuzzy silhouette widths require membership probabilities. Defaults to `NULL`.

`a` Non-negative number, the exponent α of the weights in fuzzy silhouette averaging. Higher values increase the emphasis on strong membership differences; `a = 0` gives every observation the same weight, so the fuzzy average equals the crisp average. Defaults to `2`.

`sort` Logical; if `TRUE`, sorts the output `widths` data frame by cluster and descending silhouette width. Defaults to `FALSE`.

`print.summary` Logical; if `TRUE`, reports a summary table of average silhouette widths and sizes for each cluster through `message`. Defaults to `FALSE` for `Silhouette()` and `TRUE` for `summary()`.

`clust_fun` Optional S3 or S4 function object or function as character string specifying a clustering function that produces the proximity measure matrix. For example, `fcm` or `"fcm"`. If provided, `prox_matrix` must be the name of the matrix component in the clustering output (e.g., `"d"` for `fcm` when `proximity_type = "dissimilarity"`). Defaults to `NULL`.

`...` Additional arguments passed to `clust_fun`, such as `x,centers` for `fcm`.

`cluster` Optional integer vector of length n giving the cluster of each observation as a column number of `prox_matrix` $(1, \ldots, K)$. Use it to score a partition produced elsewhere, for example by hierarchical clustering or by a clustering algorithm whose labels differ from the nearest representative. If `clust_fun` is provided, `cluster` may be the name of the label component of its output (e.g., `"cluster"` for `kmeans`). Must be named in the call because it follows `...`. Defaults to `NULL`.

`clust_args` A list of further arguments for `clust_fun`. Arguments given through `...` are matched first against this function's own arguments, and R matches names partially: `m = 3` meant for the fuzzifier of `fcm` would be taken as `method`. Pass such arguments here instead, e.g. `clust_args = list(m = 3)`. Defaults to `list()`.

x	An object of class "Silhouette", typically the output of the Silhouette, softSilhouette, dbSilhouette and cerSilhouette function. Also supports objects classes eclust, hcut, pam, clara, fanny, silhouette, or silhouette from **cluster**, **factoextra**, **drclust** packages. For these classes, explicitly call plotSilhouette() to generate the plot.
label	Logical; if TRUE, the x-axis is labeled with observation row indices from the input data and titled "Row Index". Defaults to FALSE.
summary.legend	Logical; if TRUE, prints a summary of average silhouette widths and sizes for each cluster in legend ("Cluster (Size): Width"). If FALSE, the legend shows only cluster numbers. Defaults to TRUE.
grayscale	Logical; if TRUE, the plot uses a grayscale color palette for clusters. If FALSE, uses the default or specified color palette. Defaults to FALSE.
linetype	Character or numeric value specifying the type of line to be used for the horizontal reference line indicating the average silhouette width. Accepts standard ggplot2 linetype values, such as: • Character: "solid", "dashed", "dotted", "dotdash", "longdash", "twodash". • Numeric: a single integer from 0 to 6 corresponding to ggplot2 line patterns. Defaults to "dashed".
object	An object of class "Silhouette", typically the output of the Silhouette, softSilhouette, dbSilhouette, and cerSilhouette function.

Details

Silhouette() is not the same as silhouette from the **cluster** package: that function needs all $n(n-1)/2$ pairwise dissimilarities, whereas Silhouette() needs only the $n \times K$ proximities between observations and cluster representatives (centroids, medoids or prototypes). The two give different, although usually similar, values.

The Silhouette function implements the Simplified Silhouette method introduced by Van der Laan, Pollard, & Bryan (2003), which adapts and generalizes the classic silhouette method of Rousseeuw (1987).

Clustering quality is evaluated using a proximity matrix, denoted as $\Delta = [\delta_{ik}]_{n \times K}$ for dissimilarity measures or $\Delta' = [\delta'_{ik}]_{n \times K}$ for similarity measures. Here, $i = 1, \ldots, n$ indexes observations, and $k = 1, \ldots, K$ indexes clusters. δ_{ik} represents the dissimilarity (e.g., distance) between observation i and cluster k, while δ'_{ik} represents similarity values.

Let c_i be the cluster of observation i (see *Cluster labels* below) and ν_i its neighboring cluster, the closest cluster other than c_i. The silhouette width $S(x_i)$ for observation i depends on the proximity type:

For **dissimilarity** measures:

$$S(x_i) = \frac{\delta_{i\nu_i} - \delta_{ic_i}}{N(x_i)}$$

For **similarity** measures:

$$S(x_i) = \frac{\delta'_{ic_i} - \delta'_{i\nu_i}}{N(x_i)}$$

where $N(x_i)$ is a normalizing factor defined by the method.

Choice of method: The normalizer $N(x_i)$ is selected according to the `method` argument:

- For `medoid` (Van der Laan et al., 2003):
  - Dissimilarity: $\max(\delta_{ic_i}, \delta_{i\nu_i})$
  - Similarity: $\max(\delta'_{ic_i}, \delta'_{i\nu_i})$
- For `pac` (Raymaekers & Rousseeuw, 2022):
  - Dissimilarity: $\delta_{ic_i} + \delta_{i\nu_i}$
  - Similarity: $\delta'_{ic_i} + \delta'_{i\nu_i}$

The names describe the normalization, not the algorithm. The name `"medoid"` is kept for compatibility with earlier versions and with Van der Laan et al. (2003), who introduced this width for partitioning around medoids; it divides by the larger of the two proximities and applies equally to centroids (as in k-means), prototypes, or any other cluster representative. Likewise, `"pac"` does not require a classifier. For the same labels and non-negative proximities, the PAC width equals $s/(2-|s|)$, where s is the medoid width, so the two rank observations identically and differ only in scale. These methods can also be applied to proximities derived from **classification algorithms**, to assess the separation of groups formed by a classifier.

Two equal proximities give a width of 0 (including two zero distances). An infinite proximity to the neighboring cluster gives a width of 1 and an infinite proximity to the assigned cluster gives -1.

Value

A data frame of class `"Silhouette"` with one row per observation and columns `cluster`, `neighbor`, `weight` (only when `prob_matrix` is supplied) and `sil_width`. The object includes the following attributes:

proximity_type The proximity type used (`"similarity"` or `"dissimilarity"`).

method The silhouette calculation method used (`"medoid"` or `"pac"`).

average Character — the averaging method: `"crisp"`, `"fuzzy"`, or `"median"`.

Further, `summary` returns (invisibly) a list containing:

- `clus.avg.widths`: A named numeric vector of average silhouette widths per cluster.
- `avg.width`: The overall average silhouette width.
- `sil.sum`: A data frame with columns `cluster`, `size`, and `avg.sil.width` summarizing cluster sizes and average silhouette widths.

Cluster labels

The cluster c_i of each observation is taken, in order of precedence, from

1. `cluster`, when supplied;
2. the largest membership in `prob_matrix`, when supplied;
3. the closest cluster in `prox_matrix`.

The neighboring cluster ν_i is always the closest cluster other than c_i in the proximity matrix. When the labels come from the proximity matrix itself (case 3), every width is non-negative, so a poorly assigned observation shows up only as a width near 0. When the labels come from elsewhere (cases 1 and 2), an observation that is closer to another cluster than to its own receives a negative width. To assess the partition produced by a clustering algorithm, supply that partition through `cluster` or `prob_matrix`.

Averages

If `average = "crisp"`, the **crisp silhouette index** (CS) is the mean width:

$$CS = \frac{1}{n}\sum_{i=1}^{n} S(x_i)$$

summarizing overall clustering quality.

If `average = "fuzzy"` and `prob_matrix` is provided, denoted as $\Gamma = [\gamma_{ik}]_{n\times K}$, with γ_{ik} representing the probability of observation i belonging to cluster k, the **fuzzy silhouette index** (FS) of Campello and Hruschka (2006) is:

$$FS = \frac{\sum_{i=1}^{n} w_i S(x_i)}{\sum_{i=1}^{n} w_i}, \qquad w_i = (\gamma_{i(1)} - \gamma_{i(2)})^{\alpha},$$

where $\gamma_{i(1)}$ and $\gamma_{i(2)}$ are the largest and second largest memberships of observation i, and α (the `a` argument) controls the emphasis on confident assignments ($\alpha \geq 0$; $\alpha = 0$ gives the crisp average). The weight w_i is stored in the `weight` column and does not depend on the cluster labels; it equals Equation (8) of Bhat Kapu and Kiruthika (2024) when each observation is assigned to its largest membership. If all weights are zero, the fuzzy average is undefined (`NaN`) and a warning is given.

If `average = "median"`, the median width is reported.

Missing widths are removed before averaging.

Input checks and ties

`prox_matrix` must be a numeric matrix with at least two columns and no missing values; `+Inf` is allowed (for example, $-\log 0$) but `-Inf` is not. Negative proximities are rejected, because the widths would then no longer lie in $[-1, 1]$; transform them first (for example, $(1 + r)/2$ for correlations). `prob_matrix` must be non-negative with rows summing to 1 within a tolerance of `sqrt(.Machine$double.eps)` (about 1.5×10^{-8}); rows are not renormalized.

The closest cluster, the neighboring cluster and the largest membership are found with `max.col(..., ties.method = "first")`. Exactly tied proximities or memberships are therefore resolved in favor of the column that comes first, so with exact ties the labels (though not the widths from the proximity matrix itself, which are then 0) depend on the order of the columns. Supply `cluster` to control the assignment of tied observations.

References

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65. doi:10.1016/0377-0427(87)901257

Van der Laan, M., Pollard, K., & Bryan, J. (2003). A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8), 575–584. doi:10.1080/0094965031000136012

Campello, R. J., & Hruschka, E. R. (2006). A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems*, 157(21), 2858–2875. doi:10.1016/j.fss.2006.07.006

Raymaekers, J., & Rousseeuw, P. J. (2022). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

Bhat Kapu, S., & Kiruthika. (2024). Some density-based silhouette diagnostics for soft clustering algorithms. Communications in Statistics: Case Studies, Data Analysis and Applications, 10(3-4), 221-238. doi:10.1080/23737484.2024.2408534

See Also

softSilhouette, dbSilhouette, cerSilhouette, calSilhouette, getSilhouette, is.Silhouette, plotSilhouette, silhouette

Examples

```
# Simplified silhouette of a k-means partition of the iris data
data(iris)
out <- kmeans(iris[, -5], 3)
if (requireNamespace("proxy", quietly = TRUE)) {
  d <- proxy::dist(iris[, -5], out$centers)
  silh_out <- Silhouette(d, print.summary = TRUE)
  plot(silh_out)

  # Score a partition produced elsewhere: average-linkage clustering,
  # with the distances to the centroids of its groups
  hc <- cutree(hclust(dist(iris[, -5]), "average"), 3)
  centers <- apply(iris[, -5], 2, function(v) tapply(v, hc, mean))
  silh_hc <- Silhouette(proxy::dist(iris[, -5], centers), cluster = hc)
  summary(silh_hc)
} else {
  message("Install 'proxy': install.packages('proxy')")
}

# Scree plot for optimal clusters (2 to 7)
if (requireNamespace("ppclust", quietly = TRUE)) {
  library(ppclust)
  avg_sil_width <- rep(NA,7)
  for (k in 2:7) {
    out <- Silhouette(
      prox_matrix = "d",
      proximity_type = "dissimilarity",
      prob_matrix = "u",
      clust_fun = ppclust::fcm,
      x = iris[, 1:4],
      centers = k,
      average = "fuzzy"
    )
    # Compute average silhouette width from widths
    avg_sil_width[k] <- summary(out, print.summary = FALSE)$avg.width
  }
  plot(avg_sil_width,
    type = "o",
    ylab = "Overall Silhouette Width",
    xlab = "Number of Clusters",
    main = "Scree Plot"
  )
```

```
} else {
  message("Install 'ppclust': install.packages('ppclust')")
}
```

`softSilhouette` *Calculate Silhouette Width for Soft Clustering Algorithms*

Description

Computes silhouette widths for soft clustering results by interpreting cluster membership probabilities (or their transformations) as proximity measures (Bhat Kapu and Kiruthika, 2024). Although originally designed for evaluating clustering quality within a method, this adaptation allows heuristic comparison across soft clustering algorithms using average silhouette widths.

Usage

```
softSilhouette(
  prob_matrix,
  prob_type = c("pp", "nlpp", "pd"),
  method = c("pac", "medoid"),
  average = c("crisp", "fuzzy", "median"),
  a = 2,
  sort = FALSE,
  print.summary = FALSE,
  clust_fun = NULL,
  ...,
  cluster = NULL,
  clust_args = list()
)
```

Arguments

`prob_matrix` A numeric matrix where rows represent observations and columns represent cluster membership probabilities. Rows must sum to 1. If `clust_fun` is provided, `prob_matrix` should be the name of the matrix component as a string (e.g., `"u"` for `fcm`).

`prob_type` Character string specifying the type transformation of membership matrix considered as proximity matrix in `prob_matrix`. Options are:

- `"pp"` Posterior probabilities $[\gamma_{ik}]_{n\times K}$ (non-negative, summing to 1 per row), treated as similarities (PPS, PACS).
- `"nlpp"` Negative log of posterior probabilities $[-\ln \gamma_{ik}]_{n\times K}$ (non-negative; $+\infty$ for a zero probability), treated as dissimilarities (NLPPS).
- `"pd"` Probability density: the component densities $f_k(x_i)$, treated as similarities (PDS). They are computed from the memberships by Bayes' theorem as $[\gamma_{ik}/\hat{\pi}_k]_{n\times K}$, with $\hat{\pi}_k = n^{-1}\sum_i \gamma_{ik}$; for a mixture fitted by EM this equals $f_k(x_i)/f(x_i)$ (see Details), and for other soft clustering algorithms the same ratio is used.

Defaults to "pp".

method — Character string specifying the normalization of the silhouette width. Options are "pac" (Probability of Alternative Cluster) or "medoid"; see Silhouette. Defaults to "pac".

average — Character string specifying the method for computing the average silhouette width. Options are:

- "crisp" – unweighted (simple) average.
- "fuzzy" – weighted average; the weight of an observation is the gap between its two largest memberships raised to the power a.
- "median" – median silhouette width across observations.

Defaults to "crisp".

a — Non-negative number, the exponent α of the weights in fuzzy silhouette averaging. Higher values increase the emphasis on strong membership differences; a = 0 gives every observation the same weight, so the fuzzy average equals the crisp average. Defaults to 2.

sort — Logical; if TRUE, sorts the output widths data frame by cluster and descending silhouette width. Defaults to FALSE.

print.summary — Logical; if TRUE, reports a summary table of average silhouette widths and sizes for each cluster through message. Defaults to FALSE for Silhouette() and TRUE for summary().

clust_fun — Optional S3 or S4 function object or function as character string specifying a clustering function that produces the proximity measure matrix. For example, fcm or "fcm". If provided, prox_matrix must be the name of the matrix component in the clustering output (e.g., "d" for fcm when proximity_type = "dissimilarity"). Defaults to NULL.

... — Additional arguments passed to clust_fun, such as x,centers for fcm.

cluster — Optional integer vector of length n giving the cluster of each observation as a column number of prob_matrix. If NULL (the default), each observation is assigned to the cluster with its largest membership (for "pp" and "nlpp") or its largest density (for "pd"); see *Cluster labels*. If clust_fun is provided, cluster may be the name of the label component of its output. Must be named in the call because it follows ....

clust_args — A list of further arguments for clust_fun. Arguments given through ... are matched first against this function's own arguments, and R matches names partially: m = 3 meant for the fuzzifier of fcm would be taken as method. Pass such arguments here instead, e.g. clust_args = list(m = 3). Defaults to list().

Details

With the names used by Bhat Kapu and Kiruthika (2024): prob_type = "pp" with method = "pac" is the PACS (the silhouette based on the probability of the alternative class of Raymaekers and Rousseeuw, 2022, applied to posterior probabilities), "pp" with "medoid" the PPS (posterior probability silhouette), "nlpp" with "medoid" the NLPPS (negative log of posterior probability silhouette), and "pd" with "medoid" the PDS (probability density silhouette). The certainty silhouette (CeS) and the density-based silhouette (DBS) are computed by cerSilhouette and dbSilhouette.

The membership matrix is transformed according to `prob_type` and the widths are computed as in `Silhouette`. The PPS and PACS treat the posterior probabilities as a similarity proximity measure, the NLPPS turns them into a dissimilarity through the negative logarithm, and the PDS treats the component densities as a similarity (Bhat Kapu and Kiruthika, 2024). With `method = "pac"`, the `"pp"` width is $(\gamma_{ic_i} - \gamma_{i\nu_i})/(\gamma_{ic_i} + \gamma_{i\nu_i})$; with `"medoid"` it is $1 - \gamma_{i\nu_i}/\gamma_{ic_i}$.

For a finite mixture $f(x) = \sum_k \pi_k f_k(x)$ fitted by EM, $\gamma_{ik} = \pi_k f_k(x_i)/f(x_i)$ and, at convergence, $\hat{\pi}_k$ is the fitted mixing proportion, so $\gamma_{ik}/\hat{\pi}_k = f_k(x_i)/f(x_i)$ (up to the convergence tolerance). The PDS width of Equation (22) of Bhat Kapu and Kiruthika (2024), $(f_{c_i}(x_i) - \max_{k \neq c_i} f_k(x_i))/\max(f_{c_i}(x_i), \max_{k \neq c_i} f_k(x_i))$, is therefore computed from the memberships alone, and can be used with any soft clustering algorithm.

A zero probability gives an infinite `"nlpp"` dissimilarity. The width is then 1 if only the neighboring membership is zero, -1 if only the assigned membership is zero, and 0 if both are zero.

Value

A data frame of class `"Silhouette"` with one row per observation and columns `cluster`, `neighbor`, `weight` (only when `average = "fuzzy"`) and `sil_width`. The object includes the following attributes:

proximity_type The proximity type used (`"similarity"` or `"dissimilarity"`).

method The silhouette calculation method used (`"medoid"` or `"pac"`).

average Character — the averaging method: `"crisp"`, `"fuzzy"`, or `"median"`.

Choosing the number of clusters

The average of these widths over $k = 2, 3, \ldots$ typically starts near 1 and decreases. Bhat Kapu and Kiruthika (2024) read the number of clusters from the reverse elbow drop of this curve, the value of k after which the average drops sharply, rather than from its maximum. Which k is suggested depends on the algorithm that produced the memberships, on the width and on the averaging, so report all three with the result.

Cluster labels

The labels are `cluster` when supplied. Otherwise, for `"pp"` and `"nlpp"`, each observation is assigned to the cluster with its largest membership, the partition a soft clustering algorithm reports, and the widths are non-negative. For `"pd"`, each observation is assigned to the cluster with its largest density $\gamma_{ik}/\hat{\pi}_k$, as in the definition of the PDS, so the PDS widths lie in $[0, 1]$ for every average. The largest density and the largest membership coincide when the clusters have equal estimated weights; otherwise the PDS can place an observation in a smaller cluster than its largest membership does. The neighboring cluster is the closest other cluster in the transformed matrix.

Ties are resolved in favor of the first column; see `Silhouette`.

Averages

If `average = "crisp"`, the **crisp silhouette index** (CS) is the mean width:

$$CS = \frac{1}{n} \sum_{i=1}^{n} S(x_i)$$

summarizing overall clustering quality.

If `average = "fuzzy"` and `prob_matrix` is provided, denoted as $\Gamma = [\gamma_{ik}]_{n \times K}$, with γ_{ik} representing the probability of observation i belonging to cluster k, the **fuzzy silhouette index** (FS) of Campello and Hruschka (2006) is:

$$FS = \frac{\sum_{i=1}^{n} w_i S(x_i)}{\sum_{i=1}^{n} w_i}, \qquad w_i = (\gamma_{i(1)} - \gamma_{i(2)})^{\alpha},$$

where $\gamma_{i(1)}$ and $\gamma_{i(2)}$ are the largest and second largest memberships of observation i, and α (the `a` argument) controls the emphasis on confident assignments ($\alpha \geq 0$; $\alpha = 0$ gives the crisp average). The weight w_i is stored in the `weight` column and does not depend on the cluster labels; it equals Equation (8) of Bhat Kapu and Kiruthika (2024) when each observation is assigned to its largest membership. If all weights are zero, the fuzzy average is undefined (`NaN`) and a warning is given.

If `average = "median"`, the median width is reported.

Missing widths are removed before averaging.

References

Raymaekers, J., & Rousseeuw, P. J. (2022). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

Campello, R. J., & Hruschka, E. R. (2006). A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems*, 157(21), 2858–2875. doi:10.1016/j.fss.2006.07.006

Bhat Kapu, S., & Kiruthika. (2024). Some density-based silhouette diagnostics for soft clustering algorithms. Communications in Statistics: Case Studies, Data Analysis and Applications, 10(3-4), 221-238. doi:10.1080/23737484.2024.2408534

See Also

`Silhouette`, `dbSilhouette`, `cerSilhouette`, `calSilhouette`, `getSilhouette`, `is.Silhouette`, `plotSilhouette`

Examples

```
# A small membership matrix
prob <- matrix(c(0.90, 0.10,
                 0.60, 0.40,
                 0.20, 0.80,
                 0.45, 0.55), ncol = 2, byrow = TRUE)
softSilhouette(prob)
softSilhouette(prob, prob_type = "nlpp", average = "fuzzy")


# Compare two soft clustering algorithms on the iris data
data(iris)
if (requireNamespace("ppclust", quietly = TRUE)) {
  fcm_result <- ppclust::fcm(iris[, 1:4], 3)
  out_fcm <- softSilhouette(prob_matrix = fcm_result$u, print.summary = TRUE)
  plot(out_fcm)
```

```
  fcm2_result <- ppclust::fcm2(iris[, 1:4], 3)
  out_fcm2 <- softSilhouette(prob_matrix = fcm2_result$u)
  c(fcm = summary(out_fcm, print.summary = FALSE)$avg.width,
    fcm2 = summary(out_fcm2, print.summary = FALSE)$avg.width)
} else {
  message("Install 'ppclust' to run this example: install.packages('ppclust')")
}
```

Index

calSilhouette, 2, *8*, *12*, *27*, *31*
cerSilhouette, *4, 5*, 6, *12*, *14*, *16*, *18–20*, *24*, *27*, *29*, *31*
clara, *19*, *24*

dbSilhouette, *4, 5*, *8*, 9, *14*, *16*, *18–20*, *24*, *27*, *29*, *31*

eclust, *19*, *24*
extSilhouette, 13

fanny, *19*, *24*
fcm, *3–5*, *7*, *10*, *22, 23*, *28, 29*
format.extSilhouette *(*extSilhouette*)*, 13
fviz_silhouette, *20*

get_palette, *19*
getSilhouette, *8*, *12*, *14*, 15, *18*, *20*, *27*, *31*
ggpar, *19*

hcut, *19*, *24*

is.Silhouette, *8*, *12*, *14*, *16*, 17, *20*, *27*, *31*

kmeans, *23*

max.col, *26*
message, *3*, *7*, *10*, *13*, *23*, *29*

pam, *19*, *24*
plot.Silhouette *(*Silhouette*)*, 21
plotSilhouette, *8*, *12*, *16*, 18, *18*, *27*, *31*
print.extSilhouette *(*extSilhouette*)*, 13

Silhouette, *4, 5*, *8*, *11–14*, *16*, *18–20*, 21, *24*, *29–31*
silhouette, *19*, *24*, *27*
softSilhouette, *4, 5*, *7, 8*, *12–14*, *16*, *18–20*, *24*, *27*, 28
summary.Silhouette *(*Silhouette*)*, 21