

# Package ‘FuzzySpec’

May 7, 2026

**Type** Package

**Title** Fuzzy Spectral Clustering with Variable-Weighted Adjacency Matrices

**Version** 1.0.0

**Date** 2025-09-23

**Author** Jesse S. Ghashti [aut, cre],  
John R.J. Thompson [aut]

**Maintainer** Jesse S. Ghashti <jesse.ghashti@ubc.ca>

## Description

Implementation of the FVIBES, the Fuzzy Variable-Importance Based Eigenspace Separation algorithm as described in the paper by Ghashti, J.S., Hare, W., and J.R.J. Thompson (2025). Variable-Weighted Adjacency Constructions for Fuzzy Spectral Clustering. Submitted.

**License** GPL-2

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

**Imports** stats, Thresher, fclust, mclust, mvtnorm, np, ggplot2,  
viridisLite

**Suggests** knitr, MASS, rmarkdown, patchwork, devtools, spelling

**VignetteBuilder** knitr

**NeedsCompilation** no

**RoxygenNote** 7.3.3

**Language** en-US

**Repository** CRAN

**Date/Publication** 2025-10-08 20:00:03 UTC

## Contents

clustering.accuracy . . . . .	2
compute.sigma . . . . .	3
compute.SNN . . . . .	4

fari . . . . .	5
find.radius . . . . .	6
fuzzy.spectral.clustering . . . . .	7
gen.fuzzy . . . . .	10
make.adjacency . . . . .	12
plot.fuzzy . . . . .	15
rNN.dist . . . . .	17
<b>Index</b>	<b>18</b>

---

clustering.accuracy     *Clustering Accuracy with Optimal Label Matching*

---

## Description

Computes the fraction of correctly classified observations between two label vectors after optimally matching cluster labels using `Thresher::matchLabels`.

## Usage

```
clustering.accuracy(A, B)
```

## Arguments

A                    An integer or character vector of cluster labels of length  $n$ .  
 B                    An integer or character vector of cluster labels of length  $n$ .

## Details

The function creates the contingency table `table(A, B)`, permutes columns to best align labels using `matchLabels`, and returns the sum of the diagonal divided by  $n$ .

Inputs must have equal length and the same number of unique labels; otherwise an error is given.

## Value

A single numeric value in  $[0, 1]$ : the accuracy after optimal label matching.

## References

K. R. Coombes (2025). *Thresher: Threshing and Reaping for Principal Components*. R package version 1.1.5.

## See Also

[fuzzy.spectral.clustering](#), [gen.fuzzy](#), [plot.fuzzy](#), [matchLabels](#)

**Examples**

```

set.seed(1)
n <- 200
k <- 3
A <- sample.int(k, n, replace = TRUE) # assumed true clustering labels
perm <- sample.int(k) # assumed predicted labels (sampled by permutating)
B <- perm[A]
flips <- sample.int(n, 20) # add some error a few errors
B[flips] <- sample.int(k, length(flips), replace = TRUE)

clustering.accuracy(A, B)

```

---

compute.sigma	<i>Compute Locally-Adaptive Scaling Parameters from a Distance Matrix</i>
---------------	---

---

**Description**

Derives pointwise scale parameters  $\sigma_i$  from a distance matrix, based on the  $r$ -th nearest neighbour distances. This is useful for constructing the adaptive similarity graphs proposed by Zelnik-Manor and Perona (2004).

**Usage**

```
compute.sigma(distance, r = NULL)
```

**Arguments**

distance	An $n \times n$ numeric (symmetric) distance matrix. Required.
r	Integer valued neighbourhood radius. If NULL, this value is estimated adaptively with <a href="#">find.radius</a> .

**Details**

For each observation  $i$ , the function sorts the distances `distance[i, ]`, excludes the zero self-distance, and takes the  $(r + 1)$  smallest value, where  $\sigma_i$  reflects the distance to the  $r$ -th nearest neighbour for observation  $i$ .

**Value**

A list with components:

sigma	A numeric vector of length $n$ , containing local scale parameters.
radius	The neighborhood radius $r$ used.

## References

Ghashti, J. S., Hare, W., and J. R. J. Thompson (2025). Variable-weighted adjacency constructions for fuzzy spectral clustering. Submitted.

Zelnik-Manor, L. and P. Perona (2004). Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 17.

## See Also

[make.adjacency](#), [gen.fuzzy](#), [plot.fuzzy](#), [rNN.dist](#), [find.radius](#), [compute.SNN](#), [fuzzy.spectral.clustering](#)

## Examples

```
set.seed(1)
X <- matrix(rnorm(50), nrow = 10)
D <- as.matrix(dist(X))

res <- compute.sigma(D) # automatically determine r
res$sigma
res$radius

res2 <- compute.sigma(D, r = 3) # user-specified r
res2$sigma
```

---

compute.SNN

*Shared-Nearest-Neighbours (SNN) Similarity from a Similarity Matrix*

---

## Description

Builds a Shared-Nearest-Neighbours (SNN) similarity matrix from an input similarity matrix `similarity`. For each pair of observations, the SNN score is the fraction of shared indices among their top- $r$  neighbour lists.

## Usage

```
compute.SNN(similarity, r)
```

## Arguments

<code>similarity</code>	An $n \times n$ numeric <i>similarity</i> matrix. The diagonal is assumed to correspond to self-similarity and is ignored when forming neighbour lists.
<code>r</code>	Integer value number of nearest neighbours per observation used to compute SNN overlap.

### Details

For each observation  $i$ , the function forms its neighbour set by ordering `similarity[i, ]` in decreasing order, dropping  $i$  itself, and retaining the first  $r$  indices. For a pair  $(i, j)$ , the SNN similarity is

$$\text{SNN}(i, j) = \frac{|N_r(i) \cap N_r(j)|}{r},$$

i.e., the size of the intersection of their neighbour sets divided by  $r$ . The result is symmetric with ones on the diagonal.

### Value

An  $n \times n$  symmetric numeric matrix `SNN.S` with entries in  $[0, 1]$ .

### References

Ghashti, J. S., Hare, W., and J. R. J. Thompson (2025). Variable-weighted adjacency constructions for fuzzy spectral clustering. Submitted.

Jarvis, R. A., and A. E. Patrick (1973). Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 22(11), 1025-1034.

### See Also

[make.adjacency](#), [gen.fuzzy](#), [plot.fuzzy](#), [rNN.dist](#), [find.radius](#), [compute.sigma](#), [compute.SNN](#), [fuzzy.spectral.clustering](#)

### Examples

```
set.seed(1)
X <- matrix(rnorm(50), nrow = 10)
D <- as.matrix(dist(X))
r <- 3
S <- exp(-D^2)
SNN <- compute.SNN(S, r)
head(SNN, 5)

# inspect average SNN similarity to nearest neighbour
rowMeans(SNN - diag(diag(SNN)))
```

### Description

Computes fuzzy generalizations of the Adjusted Rand Index based on Frobenius inner products of membership matrices. These measures extends the Adjusted Rand Index to compare fuzzy partitions.

**Usage**

```
fari(a, b)
```

**Arguments**

**a** An  $n \times G_1$  matrix of hard or fuzzy cluster memberships, where each row sums to 1.

**b** An  $n \times G_2$  matrix of hard or fuzzy cluster memberships, where each row sums to 1.

**Value**

A single numeric value

**fari** The Frobenius Adjusted Rand index between a and b.

**References**

Andrews, J.L., Browne, R. and C.D. Hvingelby (2022). On Assessments of Agreement Between Fuzzy Partitions. *Journal of Classification*, 39, 326–342.

J.L. Andrews, FARI (2013). GitHub repository, <https://github.com/its-likeli-jeff/FARI>

**Examples**

```
set.seed(1)
a <- matrix(runif(600), nrow = 200, ncol = 3)
a <- a / rowSums(a)
b <- matrix(runif(600), nrow = 200, ncol = 3)
b <- b / rowSums(b)

fari(a, b)
```

---

find.radius

*Adaptive Radius Selection by Natural Neighbours*

---

**Description**

Implements a Natural Neighbour search to adaptively determine a neighbourhood radius  $r$  from a general distance matrix. The algorithm increases  $r$  until the number of points with zero in-degree in the  $r$ -nearest-neighbour graph no longer decreases, and returns this radius  $r$ .

**Usage**

```
find.radius(D)
```

**Arguments**

**D** An  $n \times n$  numeric (symmetric) distance matrix.

## Details

This procedure is adapted from the *Natural Neighbor (NaN)* algorithm by Zhu, Feng and Huang (2016). The algorithm works as follows:

1. For each integer  $r$ , build the directed  $r$ -nearest-neighbour graph from  $D$ .
2. Compute the in-degree counts, i.e., how many times each observation appears among others' first  $r$  neighbours.
3. Track the number of zero in-degree points; if this number stops decreasing when  $r$  increases, the algorithm stops and returns the current  $r$ , interpreted as the natural neighbour radius.

This provides a parameter-free way to adaptively set the neighbourhood size.

## Value

A single integer  $r$ , the selected neighbourhood radius.

## References

Zhu, Q., Feng, J., and J. Huang (2016). Natural neighbor: A self-adaptive neighborhood method without parameter  $K$ . *Pattern recognition letters*, 80, 30-36.

## See Also

[make.adjacency](#), [gen.fuzzy](#), [plot.fuzzy](#), [rNN.dist](#), [compute.sigma](#), [compute.SNN](#), [fuzzy.spectral.clustering](#)

## Examples

```
set.seed(1)
X <- matrix(rnorm(100), nrow = 20)
D <- as.matrix(dist(X))
r <- find.radius(D) # Estimate the natural neighbour radius
r
rNN.dist(D, r) # use selected r for rNN.dist
```

---

fuzzy.spectral.clustering

*Fuzzy Spectral Clustering with Normalized Eigenvectors*

---

## Description

Implementation of the FVIBES algorithm by Ghashti, Hare, and Thompson (2025). Performs spectral clustering on a similarity (adjacency) matrix and returns either fuzzy  $c$ -means memberships or Gaussian mixture posterior probabilities computed on the leading normalized eigenvectors.

**Usage**

```
fuzzy.spectral.clustering(W = NULL,
                          k = NULL,
                          m = NULL,
                          method = "CM",
                          nstart = 10,
                          max.iter = 1000)
```

**Arguments**

W	A nonnegative $n \times n$ similarity (adjacency) matrix. Diagonal entries are set to 0 internally. Required.
k	Integer number of clusters. Required.
m	Fuzzy parameter for c-means, only used when method = "CM". When not provided, algorithm will set $m = 2$ .
method	Clustering method applied to the spectral embedding with "CM" for fuzzy c-means with <b>fclust</b> , or "GMM" for Gaussian mixtures with <b>mclust</b> . Default is "CM".
nstart	Number of random starts for <b>fclust::FKM</b> when method = "CM".
max.iter	Maximum number of iterations for <b>fclust::FKM</b> when method = "CM".

**Details**

Let  $D$  be the diagonal degree matrix with  $D_{ii} = \sum_j W_{ij}$ . The routine forms the symmetrically normalized similarity  $L = D^{-1/2}WD^{-1/2}$ , (Ng, Jordan, and Weiss, 2001) computes its top  $k$  eigenvectors, stacks them in  $X \in \mathbb{R}^{n \times k}$ , and row-normalizes to  $Y$  with  $Y_{i \cdot} = X_{i \cdot} / \|X_{i \cdot}\|_2$ . Clustering is then performed in the rows of  $Y$ .

When method = "CM", clustering uses c-means (Bezdek, 1981) with **fclust::FKM** on  $Y$  with fuzzy parameter  $m$ , number of starts `nstart`, and maximum iterations `max.iter`. When method = "GMM", clustering uses Gaussian mixture models (see McLachlan and Krishnan, 2008) with **mclust::Mclust** with  $G = k$  on  $Y$ .

**Value**

A list with components:

cluster	An integer vector of length $n$ hard cluster labels.
u	An $n \times k$ matrix of fuzzy cluster memberships: for "CM", fuzzy c-means memberships $U$ ; for "GMM", posterior probabilities $Z$ .
evecs	The $n \times k$ matrix $Y$ of row-normalized leading eigenvectors, i.e., the spectral embedding.
centers	Cluster centers for the embedding matrix $Y$ .

## References

- J.C. Bezdek (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.
- Ferraro, M.B., Giordani, P., and A. Serafini (2019). fclust: An R Package for Fuzzy Clustering. *The R Journal*, 11.
- Ghashti, J. S., Hare, W., and J. R. J. Thompson (2025). Variable-weighted adjacency constructions for fuzzy spectral clustering. Submitted.
- McLachlan, G. and T. Krishnan (2008). *The EM algorithm and extensions*, Second Edition. John Wiley & Sons.
- Ng, A., Jordan, M., and Y. Weiss (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14.
- Scrucca, L., Fraley, C., Murphy, T.B., and A. E. Raftery (2023). *Model-Based Clustering, Classification, and Density Estimation Using mclust in R*. Chapman & Hall.

## See Also

[make.adjacency](#), [gen.fuzzy](#), [plot.fuzzy](#), [rNN.dist](#), [find.radius](#), [compute.sigma](#), [compute.SNN](#), [npudensbw](#), [FKM](#), [Mclust](#)

## Examples

```
set.seed(1)
d <- gen.fuzzy(n = 300,
              dataset = "spirals",
              noise = 0.18)

plot.fuzzy(d) # visualize data generating process

adj <- make.adjacency(data = d$X,
                    method = "vw",
                    isLocWeighted = TRUE,
                    isModWeighted = FALSE,
                    isSparse = FALSE,
                    ModMethod = NULL,
                    scale = FALSE,
                    sig = 1,
                    radius = NULL,
                    cv.method = "cv.ls") # vwla-id from paper

spectRes <- fuzzy.spectral.clustering(W = adj,
                                     k = 3,
                                     m = 1.5,
                                     method = "CM",
                                     nstart = 50,
                                     max.iter = 1000)

head(spectRes$u) # first 6 rows of U
```

```

plotDf <- list(
  X = d$X,
  y = factor(spectRes$cluster),
  U = spectRes$u,
  k = 3
)

plot.fuzzy(plotDf) # visualize results

clustering.accuracy(d$y, spectRes$cluster) # compare results

```

---

gen.fuzzy

*Generate 2D synthetic datasets with known fuzzy memberships*


---

## Description

Simulates several 2D datasets together with fuzzy cluster memberships  $U$ . The memberships are defined by analytic density/curve proximity rules (detailed below) so they can be used as "ground truth" for fuzzy clustering and visualization (see [plot.fuzzy](#)).

## Usage

```

gen.fuzzy(n = 500,
  dataset = c("gaussian", "hyperbolas", "spirals",
             "wedges", "rings", "worms", "random"),
  k = NULL,
  noise = 0.1,
  covType = c("spherical", "diagonal", "rotated", "correlated"),
  seed = NULL)

```

## Arguments

n	Total number of observations.
dataset	Which data generator to use, with options "gaussian", "hyperbolas", "spirals", "wedges", "rings", "worms", or "random".
k	Number of clusters for dataset="random", ignored otherwise; if NULL, defaults to k = 20.
noise	Additive noise or curve-thickness parameter for applicable generators (see Details).
covType	Covariance structure for dataset="random"; one of "spherical", "diagonal", "rotated", "correlated".
seed	Optional seed for reproducibility.

## Details

Let  $X \in \mathbb{R}^{n \times 2}$  be the simulated observations and  $U \in \mathbb{R}^{n \times k}$  the fuzzy memberships. For each dataset, memberships are defined below and row-normalized to sum to 1.

**gaussian (k = 3).** Three Gaussian components with means  $(-2, 0)$ ,  $(2, 0)$ ,  $(0, 3)$  and covariances  $([1, 0.3]; [0.3, 1])$ ,  $([1, -0.3]; [-0.3, 1])$  and  $([0.8, 0]; [0, 0.8])$ . If component sizes are  $\pi_j$ , then  $U_{ij} \propto \pi_j \phi_2(x_i | \mu_j, \Sigma_j)$ .

**hyperbolas (k = 5).** One Gaussian near  $(0, 0)$  and four hyperbola branches  $\{(x, y) : (x \pm a)^2/b^2 - (y)^2/a^2 = 1\}$  and its rotated or flipped analogues, sampled along  $t \in [-2, 2]$  with noise. For observation  $x_i$ ,  $w_{\text{ball}} = 50 \cdot \phi_2(x_i | (0, 0), 0.2I_2)$ , and  $w_{\text{hyp}, \ell} = \exp(-d^2(x_i, \mathcal{C}_\ell)/(\sigma^2))$ , where  $d(\cdot, \mathcal{C}_\ell)$  is minimum distance to branch  $\ell$  for curve  $\mathcal{C}$ . We set  $U_{i \cdot} \propto w$ .

**spirals (k = 3).** Three spirals generated by  $(r, \theta) \mapsto (x, y) = ((0.5 + 0.8t) \cos(\theta_s + t), (0.5 + 0.8t) \sin(\theta_s + t))$  with shifts  $\theta_s \in \{0, 2\pi/3, 4\pi/3\}$ , with additive noise. For each spiral  $s$ ,  $d_s = \min_{t \in [0, \pi]} \|x_i - \gamma_s(t)\|$ , where  $\gamma_s(t)$  is the parameterized spiral curve described above, and  $U_{is} \propto \exp(-d_s^2/\sigma^2)$ . Note, if  $\|x_i\| < 1$ , set  $U_{i \cdot} \leftarrow (1 - \alpha)U_{i \cdot} + \alpha(1, 1, 1)/3$  with  $\alpha = 0.5e^{-\|x_i\|}$  and normalize after.

**wedges (k = 8).** Eight angular wedges with inner/outer radii 1 and 4, respectively, with small gaps between wedges. For observation  $x_i$  with radius  $r$  and angle  $\theta$ , membership to wedge  $j$  is  $U_{ij} \propto \exp(-\delta(\theta, \theta_j)^2/\sigma^2)$ , where  $\delta$  is a wrapped angular distance to the wedge centre angle  $\theta_j$ .

**rings (k = 3).** For  $x_i \in \mathbb{R}^2$  and  $r_i = \|x_i\|_2$ , there are three concentric rings with radii  $R_j \in \{1, 2.5, 4\}$  with widths  $W_j \in \{0.3, 0.4, 0.5\}$  for  $j = 1, 2, 3$ . Let  $w_{ij} = \exp(-(r_i - R_j)^2/W_j^2)$ , then  $U_{ij} = w_{ij} / \sum_{\ell=1}^3 w_{i\ell}$ .

**worms (k = 4).** Each worm  $j$  is a sinusoidal curve parameterized on  $t \in [0, 2\pi]$  by  $\gamma_j(t) = (x(t), y_j(t))$  with  $x(t) = 2(t - \pi)$ ,  $y_j(t) = A_j \sin(f_j t + \phi_j) + y_j^{\text{off}}$ , with amplitudes  $A_j$ , frequencies  $f_j$ , phases  $\phi_j$ , and vertical offsets  $y_j^{\text{off}}$ . For observation  $x_i \in \mathbb{R}^2$ , the distance to worm  $j$  is  $d_j(x_i) = \min_{t \in [0, 2\pi]} \|x_i - \gamma_j(t)\|_2$ . Then  $w_{ij} = \exp(-d_j(x_i)^2/\sigma^2)$ , and  $U_{ij} = w_{ij} / \sum_{\ell=1}^4 w_{i\ell}$ .

**random (k is user-specified).** Mixture of  $k$  Gaussians with common covariance determined by `covType` with random centres in  $[0, 30]^2$  and random cluster sizes. With mixture weights  $\pi_j$ ,  $U_{ij} \propto \pi_j \phi_2(x_i | \mu_j, \Sigma)$ .

## Value

A list with components:

X	An $n \times 2$ numeric matrix of observations.
U	An $n \times k$ matrix of probabilistic/fuzzy cluster memberships.
y	A vector length $n$ of integers corresponding to hard cluster labels.
k	Number of clusters.
centres, clusSz, covMatrix	Returned only for <code>dataset="random"</code> : the centres, cluster sizes, and common covariance used.

## Notes

The noise argument is used by "gaussian", "hyperbolas", "spirals", "rings", and "worms"; it is ignored by "wedges".

**See Also**[plot.fuzzy](#)**Examples**

```

set.seed(1)

g <- gen.fuzzy(n = 600, dataset = "gaussian", seed = 1)
plot.fuzzy(g, plotFuzzy = TRUE, colorCluster = TRUE)

s <- gen.fuzzy(n = 450, dataset = "spirals", noise = 0.2, seed = 1)
plot.fuzzy(s, plotFuzzy = TRUE, colorCluster = FALSE)

r <- gen.fuzzy(n = 800, dataset = "random", k = 15, covType = "rotated", seed = 1)
plot.fuzzy(r, plotFuzzy = TRUE, colorCluster = TRUE)

```

make.adjacency

*A General Framework for Adjacency Matrix Construction***Description**

Builds an  $n \times n$  adjacency matrix from data using Euclidean or variable-weighted distances, with optional locally-adaptive scalings and optional variable weighting by shared-nearest-neighbours (SNN), similarity ranks (SIM), or both. The options reproduce a family of adjacency matrices including the variants described by Ghashti, Hare and Thompson (2025).

**Usage**

```

make.adjacency(data,
               method = "vw",
               isLocWeighted = FALSE,
               isModWeighted = FALSE,
               isSparse = FALSE,
               ModMethod = NULL,
               scale = FALSE,
               sig = 1,
               radius = NULL,
               cv.method = "cv.ls")

```

**Arguments**

data	Numeric matrix or data frame of size $n \times p$ . Required.
method	Distance construction "eu" for Euclidean; "vw" for variable-weighted scaling (see Details). Defaults to "v".
isLocWeighted	Logical. If TRUE, use locally-adaptive scalings $\sigma_i$ (Zelnik–Perona) to self-tune the kernel; otherwise use a global scale sig. Defaults to FALSE.

isModWeighted	Logical. If TRUE, apply a weighting matrix $M$ based on SNN and/or SIM (see Details). Defaults to FALSE
isSparse	Logical. If TRUE and isModWeighted = TRUE, then SNN (ModMethod = "snn") and/or SIM (ModMethod = "sim") arguments creates a sparse adjacency matrix (see Details). Defaults to FALSE.
ModMethod	One of "snn", "sim", or "both" when isModWeighted=TRUE.
scale	Logical; standardize columns of data before distance construction.
sig	Positive numeric value for global kernel width, used only when isLocWeighted=FALSE.
radius	Integer $r$ . If NULL, $r$ is estimated with <code>find.radius</code> on the constructed distance matrix.
cv.method	Bandwidth selector for method="vw" passed to <code>np: : npudensbw</code> ; one of "cv.ml" or "cv.ls".

## Details

### Step 1: Distance.

- method="eu":  $D_{ij} = \|x_i - x_j\|_2$ .
- method="vw": compute product-kernel bandwidths  $h$  via `np: : npudensbw`, set feature-weights  $w_j = 1/h_j^2$ , rescale data as  $\tilde{x}_{ij} = \sqrt{w_j} x_{ij}$ , then  $D_{ij} = \|\tilde{x}_i - \tilde{x}_j\|_2$ . (Variable-weighted metric.)

### Step 2: Similarity kernel.

- *Locally-adaptive scaling* from Zelnik-Manor: if isLocWeighted=TRUE, compute  $\sigma_i$  as the distance to the  $r$ -th neighbour with `compute.sigma` and set

$$S_{ij} = \exp\left(-D_{ij}^2/(\sigma_i\sigma_j)\right), \quad S_{ii} = 1.$$

- *Global scale*: if isLocWeighted=FALSE,

$$S_{ij} = \exp\left(-D_{ij}^2/\sigma^2\right), \quad S_{ii} = 1.$$

### Step 3: Weighting Matrix (optional).

Let  $\text{SNN}_{ij}$  be the shared- $r$ -NN overlap fraction (see `compute.SNN`) for observations  $i$  and  $j$ , and  $\rho_i$  be the  $(r+1)$ -largest entry of observation  $i$  in matrix  $S$ . Define  $\text{SIM}_{ij} = \sqrt{\rho_i\rho_j}$ . For isModWeighted=TRUE we have the following options

- ModMethod="snn":  $M_{ij} = \begin{cases} 0.5(1 + \text{SNN}_{ij}), & \text{if isSparse=FALSE,} \\ \text{SNN}_{ij}, & \text{if isSparse=TRUE.} \end{cases}$
- ModMethod="sim":  $M_{ij} = \begin{cases} 0.5(1 + \text{SIM}_{ij}), & \text{if isSparse=FALSE,} \\ \text{SIM}_{ij}, & \text{if isSparse=TRUE.} \end{cases}$
- ModMethod="both":  $M_{ij} = \begin{cases} 0.25(1 + \text{SIM}_{ij})(1 + \text{SNN}_{ij}), & \text{if isSparse=FALSE,} \\ \text{SIM}_{ij} \cdot \text{SNN}_{ij}, & \text{if isSparse=TRUE.} \end{cases}$

The returned adjacency is  $W = S \circ M$  when isModWeighted = TRUE, otherwise  $W = S$ . These choices align with the table of named adjacency matrices (see Mapping below).

**Value**

An  $n \times n$  numeric adjacency matrix  $W$  with ones on the diagonal.

**Mapping to named adjacency matrices**

Relating to the paper by Ghashti, Hare, and Thompson (2025), let "vw" denote the variable-weighted distance method="vw" and "eu" for the traditional squared Euclidean distance; "la" denotes locally-adaptive scaling when isLocWeighted=TRUE (Zelnik-Manor and Perona, 2004); "id" denotes identity for isModWeighted = FALSE, and "sim", "snn" and "simsnn" denote weightings for  $M$  described above.

To reproduce results from the 2025 paper, below are a few examples of adjacency construction:

- vw-id:  $\exp(-D^2/\sigma^2)$  with method="vw", isLocWeighted=FALSE, isModWeighted=FALSE.
- vwla-id:  $\exp(-D^2/(\sigma_i\sigma_j))$  with method="vw", isLocWeighted=TRUE, isModWeighted=FALSE.
- vw-sim:  $0.5 \exp(-D^2/\sigma^2) (1+\text{SIM})$  with method="vw", isLocWeighted=FALSE, isModWeighted=TRUE, ModMethod="sim", isSparse=FALSE.
- vw-snns:  $0.5 \exp(-D^2/\sigma^2) (1+\text{SNN})$  with method="vw", isLocWeighted=FALSE, isModWeighted=TRUE, ModMethod="snn", isSparse=TRUE.
- vw-simsnns:  $0.25 \exp(-D^2/\sigma^2) (1+\text{SIM})(1+\text{SNN})$  with method="vw", isLocWeighted=FALSE, isModWeighted=TRUE, ModMethod="both", isSparse=TRUE.

Also note that:

- If radius is NULL,  $r$  is chosen adaptively via `find.radius` on the constructed distance matrix.
- method="vw" requires `npudensbw` for variable weighted bandwidths, with default `np: npudensbw(data, bwmethod = cv.method, nmulti = 3)` (Hayfield and Racine, 2008).

**Notes**

- When  $r$  is determined by `find.radius`, we implement a modified version of the Natural Neighbors algorithm from Zhu et al. (2016).
- SNN is a modified version of the Shared Nearest Neighbors algorithm from Jarvis and Patrick (1973).
- More information on locally-adaptive scalings are seen in Zelnik-Manor and Perona (2004).

**References**

- Ghashti, J. S., Hare, W., and J. R. J. Thompson (2025). Variable-weighted adjacency constructions for fuzzy spectral clustering. Submitted.
- Hayfield, T., and J. S. Racine (2008). Nonparametric Econometrics: The np Package. *Journal of Statistical Software* 27(5).
- Jarvis, R. A., and A. E. Patrick (1973). Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 22(11), 1025-1034.
- Zelnik-Manor, L., and P. Perona (2004). Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 17.
- Zhu, Q., Feng, J., and J. Huang (2016). Natural neighbor: A self-adaptive neighborhood method without parameter  $K$ . *Pattern Recognition Letters*, 80, 30-36.

**See Also**

[gen.fuzzy](#), [plot.fuzzy](#), [rNN.dist](#), [find.radius](#), [compute.sigma](#), [compute.SNN](#), [fuzzy.spectral.clustering](#), [npudensbw](#)

**Examples**

```
set.seed(1)
X <- scale(matrix(rnorm(200), 100, 2))

W1 <- make.adjacency(X,
  method = "eu",
  isLocWeighted = TRUE) # "eula-id" named adjacency

W2 <- make.adjacency(X,
  method = "vw",
  isLocWeighted = TRUE) # "vwla-id" named adjacency

# compare W(xi,xj) i,j = 1,...,5 for eu/vw pair W1 and W2
W1[1:5,1:5]
W2[1:5,1:5]

W3 <- make.adjacency(X,
  method = "eu",
  isLocWeighted = TRUE,
  isModWeighted = TRUE,
  ModMethod = "snn",
  isSparse = FALSE) # "eula-snn" named adjacency

W4 <- make.adjacency(X,
  method = "vw",
  isLocWeighted = TRUE,
  isModWeighted = TRUE,
  ModMethod = "snn",
  isSparse = FALSE) # "vwla-snn" named adjacency

# compare W(xi,xj) i,j = 1,...,5 for eu/vw pair W3 and W4
W3[1:5,1:5]
W4[1:5,1:5]
```

---

plot.fuzzy

*Plot 2D Fuzzy Data with Optional Uncertainty Sizing and Cluster Colouring*

---

**Description**

Creates a [ggplot](#) of 2D points with optional colouring by hard labels and optional observation-size mapping to fuzzy uncertainty.

**Usage**

```
## S3 method for class 'fuzzy'
plot(x, plotFuzzy = TRUE, colorCluster = TRUE, ...)
```

**Arguments**

x	A list as returned by <a href="#">gen.fuzzy</a> , containing X, U, y, and k.
plotFuzzy	Logical; if TRUE, map observation size to uncertainty $1 - \max_j U_{ij}$ .
colorCluster	Logical; if TRUE, colour points by the hard cluster label y.
...	Additional arguments (currently unused).

**Details**

The plotting aesthetics can be modified as follows:

- If plotFuzzy and colorCluster are both TRUE (default), the plot contains cluster coloured observations that are size scaled by uncertainty.
- If only plotFuzzy is TRUE, the plot contains monochrome coloured observations that are size scaled by uncertainty.
- If only colorCluster is TRUE, the plot contains cluster coloured observations with fixed size.
- If plotFuzzy and colorCluster are both FALSE, the plot is monochrome coloured observations with fixed size.

**Value**

A ggplot object.

**References**

H. Wickham (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

**See Also**

[gen.fuzzy](#), [ggplot](#)

**Examples**

```
set.seed(1)
d1 <- gen.fuzzy(n = 600, dataset = "gaussian", seed = 1)
p1 <- plot.fuzzy(d1)
p1 # default

p2 <- plot.fuzzy(d1, plotFuzzy = TRUE, colorCluster = FALSE)
p2 # only uncertainty sizing, monochrome

p3 <- plot.fuzzy(d1, plotFuzzy = FALSE, colorCluster = TRUE)
p3 # only coloured by cluster, no uncertainty sizing
```

---

`rNN.dist`*Compute r-Nearest Neighbours from a Distance Matrix*

---

**Description**

Given a symmetric distance matrix, returns the indices of the  $r$  nearest neighbours for each observation.

**Usage**

```
rNN.dist(D, r)
```

**Arguments**

<code>D</code>	An $n \times n$ numeric (symmetric) distance matrix.
<code>r</code>	Integer indicating the number of nearest neighbours to extract for each observation.

**Details**

For each row  $i$  of  $D$ , the function orders the distances  $D[i, \cdot]$ , excludes the self-distance, and returns the indices of the first  $r$  smallest distances. This provides the indices of the  $r$  nearest neighbours of observation  $i$ .

**Value**

An  $n \times r$  integer matrix, where row  $i$  contains the indices of the  $r$  nearest neighbours of observation  $i$ .

**References**

Ghashti, J. S., Hare, W., and J. R. J. Thompson (2025). Variable-weighted adjacency constructions for fuzzy spectral clustering. Submitted.

**See Also**

[make.adjacency](#), [gen.fuzzy](#), [plot.fuzzy](#), [find.radius](#), [compute.sigma](#), [compute.SNN](#), [fuzzy.spectral.clustering](#)

**Examples**

```
set.seed(1)
X <- matrix(rnorm(20), nrow = 5)
D <- as.matrix(dist(X))
rNN.dist(D, r = 2) # find 2 nearest neighbours for each row
```

# Index

- \* **accuracy**
    - clustering.accuracy, 2
  - \* **adaptive similarity**
    - compute.sigma, 3
  - \* **adjacency**
    - make.adjacency, 12
  - \* **ari**
    - fari, 5
  - \* **classification**
    - clustering.accuracy, 2
  - \* **clustering**
    - clustering.accuracy, 2
    - fuzzy.spectral.clustering, 7
    - gen.fuzzy, 10
    - plot.fuzzy, 15
  - \* **cluster**
    - fari, 5
  - \* **cmeans**
    - fuzzy.spectral.clustering, 7
  - \* **confusion matrix**
    - clustering.accuracy, 2
  - \* **data generation**
    - gen.fuzzy, 10
  - \* **distance**
    - rNN.dist, 17
  - \* **fuzzy clustering**
    - fuzzy.spectral.clustering, 7
  - \* **fuzzy**
    - fari, 5
    - fuzzy.spectral.clustering, 7
    - gen.fuzzy, 10
    - plot.fuzzy, 15
  - \* **graph**
    - fuzzy.spectral.clustering, 7
    - rNN.dist, 17
  - \* **kernel**
    - compute.sigma, 3
  - \* **nearest neighbors**
    - find.radius, 6
    - make.adjacency, 12
    - rNN.dist, 17
  - \* **nearest neighbours**
    - compute.sigma, 3
    - compute.SNN, 4
  - \* **rand index**
    - fari, 5
  - \* **shared neighbours**
    - compute.SNN, 4
  - \* **similarity**
    - make.adjacency, 12
  - \* **simulation**
    - gen.fuzzy, 10
  - \* **spectral clustering**
    - make.adjacency, 12
  - \* **spectral**
    - fuzzy.spectral.clustering, 7
  - \* **variable weighting**
    - make.adjacency, 12
  - \* **visualization**
    - plot.fuzzy, 15
- clustering.accuracy, 2  
compute.sigma, 3, 5, 7, 9, 13, 15, 17  
compute.SNN, 4, 4, 5, 7, 9, 13, 15, 17
- fari, 5  
find.radius, 3–5, 6, 9, 13–15, 17  
FKM, 9  
fuzzy.spectral.clustering, 2, 4, 5, 7, 7, 15, 17
- gen.fuzzy, 2, 4, 5, 7, 9, 10, 15–17  
ggplot, 15, 16
- make.adjacency, 4, 5, 7, 9, 12, 17  
matchLabels, 2  
Mclust, 9
- npudensbw, 9, 14, 15

plot.fuzzy, [2](#), [4](#), [5](#), [7](#), [9](#), [10](#), [12](#), [15](#), [15](#), [17](#)

rNN.dist, [4](#), [5](#), [7](#), [9](#), [15](#), [17](#)