# Package 'ANF'

December 1, 2025

**Type** Package

**Title** Affinity Network Fusion for Complex Patient Clustering

**Version** 1.32.0

**Author** Tianle Ma, Aidong Zhang

**Maintainer** Tianle Ma <tianlema@buffalo.edu>

**Description** This package is used for complex patient clustering by integrating multi-omic data through affinity network fusion.

**License** GPL-3

**VignetteBuilder** knitr

**Imports** igraph, Biobase, survival, MASS, stats, RColorBrewer

**Suggests** ExperimentHub, SNFtool, knitr, rmarkdown, testthat

**biocViews** Clustering, GraphAndNetwork, Network

**RoxygenNote** 6.0.1

**git_url** https://git.bioconductor.org/packages/ANF

**git_branch** RELEASE_3_22

**git_last_commit** 58724a9

**git_last_commit_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2025-12-01

## Contents

affinity_matrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
ANF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
eval_clu . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
kNN_graph . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
pod . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
spectral_clustering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

**Index** **7**

## `affinity_matrix` *Generate a symmetric affinity matrix based on a distance matrix using 'local' Gaussian kernel*

### Description

Generate a symmetric affinity matrix based on a distance matrix using 'local' Gaussian kernel

### Usage

```
affinity_matrix(D, k, alpha = 1/6, beta = 1/6)
```

### Arguments

| | |
|---|---|
| `D` | distance matrix (need to be a square and non-negative matrix) |
| `k` | the number of k-nearest neighbors |
| `alpha` | coefficient for local diameters. Default value: 1/6. This default value should work for most cases. |
| `beta` | coefficient for pair-wise distance. Default value: 1/6. This default value should work for most cases. |

### Value

an affinity matrix

### Examples

```
D = matrix(runif(400), nrow=20)
A = affinity_matrix(D, 5)
```

## `ANF` *Fuse affinity networks (i.e., matrices) through one-step or two-step random walk*

### Description

Fuse affinity networks (i.e., matrices) through one-step or two-step random walk

### Usage

```
ANF(Wall, K = 20, weight = NULL, type = c("two-step", "one-step"),
  alpha = c(1, 1, 0, 0, 0, 0, 0, 0), verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `Wall` | a list of affinity matrices of the same shape. |
| `K` | the number of k nearest neighbors for function kNN_graph |
| `weight` | a list of non-negative real numbers (which will be normalized internally so that it sums to 1) that one-to-one correspond to the affinity matrices included in 'Wall'. If not set, internally uniform weights are assigned to all affinity matrices in 'Wall'. |
| `type` | choose one of the two options: perform "one-step" random walk, or "two-step" random walk on the list of affinity matrices in 'Wall" to generate a fused affinity matrix. Default: "two-step" random walk |
| `alpha` | a list of eight non-negative real numbers (which will be normalized internally to make it sums to 1). Only used when "two-step" (default value of 'type') random walk is used. 'alpha' is the weights for eight terms in the "two-step" random walk formula (check research paper for more explanations about the terms). Default value: (1, 1, 0, 0, 0, 0, 0, 0), i.e., only use the first two terms (since they are most effective in practice). |
| `verbose` | logical(1); if true, print some information |

## Value

a fused transition matrix (representing a fused network)

## Examples

```
D1 = matrix(runif(400), nrow=20)
W1 = affinity_matrix(D1, 5)
D2 = matrix(runif(400), nrow=20)
W2 = affinity_matrix(D1, 5)
W = ANF(list(W1, W2), K=10)
```

---

`eval_clu` *Evaluate clustering result*

---

## Description

Evaluate clustering result

## Usage

```
eval_clu(true_class, w = NULL, d = NULL, k = 10, num_clu = NULL,
  surv = NULL, type_L = c("rw", "sym", "unnormalized"), verbose = TRUE)
```

## Arguments

| | |
|---|---|
| `true_class` | A named vector of true class labels |
| `w` | affinity matrix |
| `d` | distance matrix if w is NULL, calcuate w using d |
| `k` | an integer, default 10; if w is null, w = affinity_matrix(d, k); otherwise unused. |

| | |
|---|---|
| num_clu | an integer; number of clusters; if NULL, set num_clu to be the number of classes using true_class |
| surv | a data.frame with at least two columns: time (days_to_death or days_to_last_follow_up), and censored (logical(1)) |
| type_L | (parameter passed to spectral_clustering: ‘type‘) choose one of three versions of graph Laplacian: "unnormalized": unnormalized graph Laplacian matrix (L = D - W); "rw": normalization closely related to random walk (L = I - D^(-1)*W); (default choice) "sym": normalized symmetric matrix (L = I - D^(-0.5) * W * D^(-0.5)) For more information: https://www.cs.cmu.edu/~aarti/Class/10701/readings/Luxburg06_Tl |
| verbose | logical(1); if true, print some information |

### Value

a named list of size 3: "w": affinity matrix used for spectral_clustering; "clu.res": a named vector of calculated "NMI" (normalized mutual information), "ARI" (Adjusted Rand Index), and "-log10(p)" of log rank test of survival distributions of patient clusters; "labels: a numeric vector as class labels

### Examples

```
library(MASS)
true.class = rep(c(1,2), each=100)
feature.mat1 = mvrnorm(100, rep(0, 20), diag(runif(20,0.2,2)))
feature.mat2 = mvrnorm(100, rep(0.5, 20), diag(runif(20,0.2,2)))
feature1 = rbind(feature.mat1, feature.mat2)
d = dist(feature1)
d = as.matrix(d)
A = affinity_matrix(d, 10)
res = eval_clu(true_class=true.class, w=A)
```

## kNN_graph *Calculate k-nearest-neighbor graph from affinity matrix and normalize it as transition matrix*

### Description

Calculate k-nearest-neighbor graph from affinity matrix and normalize it as transition matrix

### Usage

```
kNN_graph(W, K)
```

### Arguments

| | |
|---|---|
| W | affinity matrix (its elements are non-negative real numbers) |
| K | the number of k nearest neighbors |

### Value

a transition matrix of the same shape as W

## Examples

```
D = matrix(runif(400),20)
W = affinity_matrix(D, 5)
S = kNN_graph(W, 5)
```

---

pod *Finding optimal discrete solutions for spectral clustering*

---

## Description

Finding optimal discrete solutions for spectral clustering

## Usage

```
pod(Y, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `Y` | a matrix with N rows and K columns, with N being the number of objects (e.g., patients), K being the number of clusters. The K columns of 'Y' should correspond to the first k eigenvectors of graph Laplacian matrix (of affinity matrix) corresponding to the k smallest eigenvalues |
| `verbose` | logical(1); if true, print some information |

## Value

class assignment matrix with the same shape as Y (i.e., N x K). Each row contains all zeros except one 1. For instance, if X_ij = 1, then object (eg, patient) i belongs to cluster j.

## References

Stella, X. Yu, and Jianbo Shi. "Multiclass spectral clustering." ICCV. IEEE, 2003.

## Examples

```
D = matrix(runif(400),20)
A = affinity_matrix(D, 5)
d = rowSums(A)
L = diag(d) - A
# `NL` is graph Laplacian of affinity matrix `A`
NL = diag(1/d) %*% L
e = eigen(NL)
# Here we select eigenvectors corresponding to three smallest eigenvalues
Y = Re(e$vectors[,-1:-17])
X = pod(Y)
```

## spectral_clustering *spectral_clustering*

### Description

spectral_clustering

### Usage

```
spectral_clustering(A, k, type = c("rw", "sym", "unnormalized"),
  verbose = FALSE)
```

### Arguments

| | |
|---|---|
| A | affinity matrix |
| k | the number of clusters |
| type | choose one of three versions of graph Laplacian: "unnormalized": unnormalized graph Laplacian matrix (L = D - W); "rw": normalization closely related to random walk (L = I - D^(-1)*W); (default choice) "sym": normalized symmetric matrix (L = I - D^(-0.5) * W * D^(-0.5)) For more information: https://www.cs.cmu.edu/~aarti/Class/1 |
| verbose | logical(1); if true, print user-friendly information |

### Value

a numeric vector as class labels

### Examples

```
D = matrix(runif(400), nrow = 20)
A = affinity_matrix(D, 5)
labels = spectral_clustering(A, k=2)
```

# Index

affinity_matrix, 2
ANF, 2

eval_clu, 3

kNN_graph, 4

pod, 5

spectral_clustering, 6