Package 'missSNF'

February 27, 2025

Type Package

Title miss-SNF: a multimodal patient similarity network integration approach to handle completely missing data sources

Version 1.0.0

Description Implementation of miss-Similarity Network Fusion (miss-SNF), a novel general-purpose data integration approach designed to manage completely missing data in the context of patient similarity networks. Miss-SNF integrates incomplete unimodal patient similarity networks by leveraging a non-linear message-passing strategy borrowed from the SNF algorithm. Miss-SNF is able to recover patient similarity even with completely missing data sources and it is "task agnostic", in the sense that can integrate partial data for both unsupervised and supervised prediction tasks.

License GPL (>= 3)

Encoding UTF-8

Imports SNFtool, NetInt, RANKS, NetPreProc, mathjaxr

RoxygenNote 7.3.2

RdMacros mathjaxr

NeedsCompilation no

Author Jessica Gliozzo [aut, cre] (<https://orcid.org/0000-0001-7629-8112>),
Giorgio Valentini [ctb] (0000-0002-5694-3919),
Elena Casiraghi [ctb] (0000-0003-2024-7572),
Mauricio A. Soto Gomez [ctb] (<https://orcid.org/0000-0001-5977-9467>)

Maintainer Jessica Gliozzo <jessica.gliozzo@unimi.it>

Contents

get.miss.pts 2
impute_miss 3
miss.snf 3
scaled.exp.chi2 6
scaled.exp.euclidean 6

Index **8**

`get.miss.pts` *Get list of missing patients for each matrix*

Description

This function takes in input a list of matrices/dataframes Mall and returns for each one the list of patients that are missing (considering the union of patients in all matrices). A patient is considered "missing" if (I) it is present in the matrix but has |NA| > perc.na, (II) a patient is not present in one matrix but it is present in at least one of the others.

Usage

```
get.miss.pts(Mall, perc.na = 0.2, miss.symbols = NULL)
```

Arguments

`Mall`	list of named matrices/dataframes (samples x features).
`perc.na`	percentage of NAs above which a patient is considered missing.
`miss.symbols`	vector of strings. If not NULL, the provided symbols in dataframes are converted to NA.

Value

list of vectors. Each vector contains the names of missing samples in a specific matrix/dataframe.

Examples

```
# Create list of input matrices
set.seed(123);
M1 <- matrix(runif(50, min = 0, max = 1), nrow = 5); # 5 samples
rownames(M1) <- paste0("ID_", 1:nrow(M1));
M1[1, 1:4] <- c(NA, NA, NA, NA);
M1[4, ] <- rep(NA, ncol(M1));

M2 <- matrix(runif(80, min = 0, max = 1), nrow = 8); # 8 samples
rownames(M2) <- paste0("ID_", 3:10);
M2[2, ] <- rep(NA, ncol(M2));
M2[4, ] <- rep(NA, ncol(M2));
M2[8, 1] <- NA;

M3 <- matrix(runif(80, min = 0, max = 1), nrow = 8); # 8 samples
rownames(M3) <- c(paste0("ID_", 1:5), paste0("ID_", 11:13));
M3[4, 1] <- NA;
M3[7, ] <- rep(NA, ncol(M3));

Mall <- list("M1"=M1, "M2"=M2, "M3"=M3);

# Call function

miss.pts <- get.miss.pts(Mall, perc.na=0.2, miss.symbols=NULL);

# If missing values are encoded with some symbols (e.g. "?"),
# the function can take this situation into account
```

```
M3[1, 3:4] <- "?"
Mall <- list("M1"=M1, "M2"=M2, "M3"=M3);

miss.pts <- get.miss.pts(Mall, perc.na=0.2, miss.symbols=c("?"));
```

`impute_miss` *Impute patients with few NAs*

Description

This function imputes missing samples using some specified method (e.g. mean, median) only for patients/samples having a percentage of NAs lower or equal to perc.na. In other words, not all missing data are imputed but only the ones for which a patient has enough data to not be considered completely missing.

Usage

```
impute_miss(data, perc.na, method, verbose = TRUE)
```

Arguments

`data`	matrix. Matrix (samples x features).
`perc.na`	numeric. Percentage of missing features.
`method`	string. An imputation method (possible options mean, median).
`verbose`	boolean. Want to print messages? (def. TRUE)

Value

Imputed data matrix.

`miss.snf` *miss-SNF: integration of patients' networks considering missing samples*

Description

Extension of the algorithm Similarity Network Fusion (https://doi.org/10.1038/nmeth.2810) able to handle the absence (complete or nearly complete) of a specific data source for a given patient by:

- "reconstruct strategy" that can partially reconstruct missing data by using information from different sources (i.e. miss-SNF ONE).
- "ignore strategy" which simply ignores missing data during the integration process (i.e. miss-SNF ZERO).
- "equidistant strategy", similar to reconstruct but sets the similarity of the partial samples with the others to a fixed value instead of zero.
- "random strategy" that sets the similatiry randomly.

Usage

```
miss.snf(
  Mall,
  sims,
  sims.arg = vector("list", length(sims)),
  mode = "reconstruct",
  perc.na = 0.2,
  miss.symbols = NULL,
  K = 20,
  t = 20,
  impute = "median",
  d = 1,
  random.walk = "none",
  p = 3,
  seed = NULL
)
```

Arguments

Mall	list of named matrices/dataframes (samples x features).
sims	vector of strings. It is a vector containing the names of the similarity measures to apply to the matrices in Mall. "scaled.exp.euclidean" is the scaled exponential euclidean distance; "scaled.exp.chi2" is the scaled exponential chi-square distance
sims.arg	list. List with the same length of "sims" where each elements is a list containing additional arguments for each similarity measure in the argument "sims". Set element to NULL if you want to use default parameters for a specific similarity measure (default).
mode	string. If you want to partially reconstruct missing data use "reconstruct" or "one" (default), otherwise ignore them during integration using "ignore" or "zero". If you use "equidistant", the self-loop for partial samples is set to 0.5 while the similarity with other samples is set to the same value so that the sample is equidistant from all other samples but more similar to itself than others. The option "random" sets randomly the similarity of partial samples (let's consider this as a baseline).
perc.na	percentage of NAs above which a patient is considered missing.
miss.symbols	vector of strings. If not NULL, the provided symbols in matrices are converted to NA.
K	Number of neighbors in K-nearest neighbors part of the algorithm.
t	Number of iterations for the diffusion process.
impute	string. Kind of imputation method to apply in case of samples with few missing values. Options are: NULL, "mean", "median". NOTE: if impute=NULL, then perc.na has to be 0 (i.e. having even one NA will make the patient treated as missing).
d	numeric. Set the diagonal of the matrix to "d" if mode="reconstruct" or mode="one" (def d=1).
random.walk	string. Use 1-step Random Walk to compute the local similarity matrix S and/or p-step Random Walk (p>=2) to compute the global similarity matrix P. random.walk=c("global", "local", "both", "none") and default is random.walk="none".

p	numeric. Number of steps for the p-step RW. Used only when global similarity matrix is computed through p-step Random Walk.
seed	numeric. Seed to get reproducible results. Needed only if mode = "random".

Value

A list with five elements:

- W : integrated similarity matrix. Note that the order of the patients is different from the order of the original matrices.
- removed.pts : vector with names of removed patients (i.e. patients present only in one matrix of Mall and having too much NAs or, more in general, if a patient is considered missing in all data sources).
- conv1 : vector with the Frobenius norm between the standard deviation across global matrices after cross-diffusion, considering consecutive steps.
- conv2 : vector containing the Frobenius norm of the difference among the integrated matrices at two consecutive steps.
- conv3 : vector containing the Frobenius norm of the difference among the integrated matrix at each iteration and the final integrated matrix (measures convergence velocity).

Examples

```
# Create list of input matrices
set.seed(123);
M1 <- matrix(runif(50, min = 0, max = 1), nrow = 5); # 5 samples
rownames(M1) <- paste0("ID_", 1:nrow(M1));
M1[1, 1:4] <- c(NA, NA, NA, NA);
M1[4, ] <- rep(NA, ncol(M1));

M2 <- matrix(runif(80, min = 0, max = 1), nrow = 8); # 8 samples
rownames(M2) <- paste0("ID_", 3:10);
M2[2, ] <- rep(NA, ncol(M2));
M2[4, ] <- rep(NA, ncol(M2));
M2[8, 1] <- NA;

M3 <- matrix(runif(80, min = 0, max = 1), nrow = 8); # 8 samples
rownames(M3) <- c(paste0("ID_", 1:5), paste0("ID_", 11:13));
M3[4, 1] <- NA;
M3[7, ] <- rep(NA, ncol(M3));

Mall <- list("M1"=M1, "M2"=M2, "M3"=M3);

# Call miss.snf using "reconstruct strategy"
W.r <- miss.snf(Mall, sims=rep("scaled.exp.euclidean", 3),
               sims.arg=list(list(kk=2), list(kk=2), list(kk=2)),
               mode="reconstruct", K=3);

# Call miss.snf using "ignore strategy"
W.i <- miss.snf(Mall, sims=rep("scaled.exp.euclidean", 3),
               sims.arg=list(list(kk=2), list(kk=2), list(kk=2)),
               mode="ignore", K=3);
```

scaled.exp.chi2 *Scaled exponential chi-square distance*

Description

It is a wrapper function to directly compute the scaled exponential chi-square distance as implemented in SNFtool library.

Usage

```
scaled.exp.chi2(M, kk = 20, sigma = 0.5)
```

Arguments

M	matrix/dataframe (samples x features).
kk	integer. Number of nearest neighbours to sparsify similarity.
sigma	numeric. Variance for local model.

Value

similarity matrix computed using scaled exponential chi-square distance.

Examples

```
# Create a matrix
set.seed(123);
M1 <- rbind(c(1,2,1,1,1,2,2), c(2,2,2,1,1,1,1), c(1,2,1,2,1,2,1),
           c(2,2,2,1,2,1,2));

rownames(M1) <- paste0("ID_", 1:nrow(M1));

# Compute similarity matrix
sim <- scaled.exp.chi2(M1, kk=3);
```

scaled.exp.euclidean *Scaled exponential euclidean distance*

Description

It is a wrapper function to directly compute the scaled exponential euclidean distance as implemented in SNFtool library.

NOTE: from SNFtool "If the data is continuous, we recommend to use the function dist2 as follows:"

dist <- (dist2(as.matrix(Data1),as.matrix(Data1)))^(1/2);

This is because the function dist2 actually computes the squared euclidean distance (euclidean distance without the square root).

Usage

```
scaled.exp.euclidean(M, kk = 20, sigma = 0.5)
```

Arguments

M	matrix/dataframe (samples x features).
kk	integer. Number of nearest neighbours to sparsify similarity.
sigma	numeric. Variance for local model.

Value

similarity matrix computed using scaled exponential euclidean distance.

Examples

```
# Create a matrix
set.seed(123);
M1 <- matrix(runif(50, min = 0, max = 1), nrow = 5); # 5 samples
rownames(M1) <- paste0("ID_", 1:nrow(M1));

# Compute similarity matrix
sim <- scaled.exp.euclidean(M1, kk=3, sigma=0.5);
```

Index

get.miss.pts, 2

impute_miss, 3

miss.snf, 3

scaled.exp.chi2, 6
scaled.exp.euclidean, 6