Package 'MetabolSSMF'

May 7, 2026

Type Package

Title Simplex-Structured Matrix Factorisation for Metabolomics Analysis

Version 0.1.0

Maintainer Wenxuan Liu <wenxuan.liu@ucdconnect.ie>

Description Provides a framework to perform soft clustering using simplex-structured matrix factorisation (SSMF). The package contains a set of functions for determining the optimal number of prototypes, the optimal algorithmic parameters, the estimation confidence intervals and the diversity of clusters. Abdolali, Maryam & Gillis, Nicolas (2020) <doi:10.1137/20M1354982>.

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports LaplacesDemon, NMF, doParallel, foreach, iterators, lsei, mclust, tidyr

Suggests knitr, rmarkdown, caroline, ggsci, BiocManager, Biobase

VignetteBuilder knitr

Depends R (>= 3.5.0)

License MIT + file LICENSE

NeedsCompilation no

Author Wenxuan Liu [aut, cre], Thomas Brendan Murphy [aut], Lorraine Brennan [aut]

Repository CRAN

Date/Publication 2025-03-05 16:10:02 UTC

Contents

`bootstrap` *Bootstrap algorithm function.*

Description

Bootstrap resampling approach to estimate the confidence intervals for the cluster prototypes.

Usage

```
bootstrap(data, k, H, mtimes = 50, lr = 0.01, ncore = 2)
```

Arguments

`data`	Data matrix or data frame.
`k`	The number of prototypes/clusters.
`H`	Matrix, input H matrix to start the algorithm. Usually the H matrix is the output of the function ssmf( ). If H is not supplied, the bootstrapped W matrix might have different prototype orders from the outputs of the function ssmf( ).
`mtimes`	Integer, number of bootstrap samples. Default number is 50.
`lr`	Optimisation learning rate in ssmf().
`ncore`	The number of cores to use for parallel execution.

Details

Create bootstrap samples of size n by sampling from the data set with replacement and repeat the steps M times. The m^{th} bootstrap sample is denoted as

$$X^{*(m)} = (x_1^{*(m)}, x_2^{*(m)}, \ldots, x_n^{*(m)}),$$

where each $x_i^{*(m)}$ is a random sample (with replacement) from the data set.

Then, apply the SSMF algorithm to each bootstrap sample and calculate the m^{th} bootstrap replicate of the prototypes matrix, which is denoted as $W^{*(m)}$.

The estimate standard deviation of M bootstrap replicates can be calculated by

$$sd(W^*) = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}[W^{*(m)} - \overline{W}^*]^2},$$

where $\overline{W}^* = \frac{1}{M}\sum_{m=1}^{M} W^{*(m)}$. Therefore, the 95% CIs for the prototypes can be calculated by

$$(\overline{W}^* - t_{(0.025,M-1)} \cdot sd(W^*),\ \overline{W}^* + t_{(0.975,M-1)} \cdot sd(W^{)}),$$

where $t_{(0.025,n-1)}$ and $t_{(0.975,n-1)}$ is the quantiles of student t distribution with 95% significance and $(M-1)$ degrees of freedom.

Value

`W.est` The W matrix estimated by bootstrap.

`lower` Lower bound of confidence intervals.

`upper` Upper bound of confidence intervals.

Author(s)

Wenxuan Liu

References

Stine, R. (1989). An Introduction to Bootstrap Methods: Examples and Ideas. Sociological Methods & Research, 18(2-3), 243-291. <doi:10.1177/0049124189018002003>

Examples

```
# example code

data <- SimulatedDataset

k <- 4

fit <- ssmf(data = data, k = k)

bootstrap(data = data , k = k, H = fit$H)
```

`diversity` *Shannon diversity index*

Description

Calculate the Shannon diversity index of the memberships of an observation. The base of the logarithm is 2.

Usage

```
diversity(x, two.power = FALSE)
```

Arguments

x	A membership vector.
two.power	Logical, whether return to the value of $2^{\mathrm{E}(h_i)}$.

Details

Given a membership vector of the i^{th} observation h_i, the Shannon diversity index is defined as

$$\mathrm{E}(h_i) = -\sum_{r=1}^{k} h_{ir} \log_2(h_{ir}).$$

Specifically, in the case of $h_{ir} = 0$, the value of $h_{ir}\log_2(h_{ir})$ is taken to be 0.

Value

A numeric value of Shannon diversity index $\mathrm{E}(h_i)$ or $2^{\mathrm{E}(h_i)}$.

Author(s)

Wenxuan Liu

Examples

```
# Memberships vector
membership1 <- c(0.1, 0.2, 0.3, 0.4)
diversity(membership1)
diversity(membership1, two.power = TRUE)

# Memberships matrix
membership2 <- matrix(c(0.1, 0.2, 0.3, 0.4, 0.3, 0.2, 0.4, 0.1, 0.2, 0.3, 0.1, 0.4),
                      nrow=3, ncol=4, byrow=TRUE)

E <- rep(NA, nrow(membership2))
for(i in 1:nrow(membership2)){
  E[i] <- diversity(membership2[i,])
}
E
```

fit_boot *Example results of bootstrap.*

Description

A list of the results for bootstrap example.

Usage

```
fit_boot
```

Format

A list of bootstrap result, including the values of estimated prototype matrix (W), the lower bound of confidence intervals and the upper bound of confidence intervals.

fit_gap *Example results of gap statistic.*

Description

A list of the results for gap statistic example for $k = 1, 2, ..., 10$.

Usage

```
fit_gap
```

Format

A list of gap statistic result, including the gap value vector, the optimal number of prototypes/clusters and the Standard error vector.

fit_SSMF *Example results of SSMF.*

Description

A list of the results for SSMF example for $k = 1, 2, ..., 10$.

Usage

```
fit_SSMF
```

Format

A list with 10 items, each item is a results of SSMF, containing the values of the estimated prototype matrix (W) and the estimated membership matrix (H) matrix and the value of the residuals sum of square (SSE).

`gap` *Gap statistic algorithm.*

Description

Estimating the number of prototypes/clusters in a data set using the gap statistic.

Usage

```
gap(
  data,
  rss,
  meth = c("kmeans", "uniform", "dirichlet", "nmf"),
  itr = 50,
  lr = 0.01,
  ncore = 2
)
```

Arguments

`data`	Data matrix or data frame.
`rss`	Numeric vector, residual sum of squares from ssmf model using the number of clusters $1, 2, \ldots, k$.
`meth`	Character, specification of method to initialise the W and H matrix, see 'method' in init( ).
`itr`	Integer, number of Monte Carlo samples.
`lr`	Optimisation learning rate in ssmf().
`ncore`	The number of cores to use for parallel execution.

Details

This gap statistic selects the biggest difference between the original residual sum of squares (RSS) and the RSS under an appropriate null reference distribution of the data, which is defined to be

$$\mathrm{Gap}(k) = \frac{1}{B}\sum_{b=1}^{B}\log(\mathrm{RSS}^*_{kb}) - \log(\mathrm{RSS}_k),$$

where B is the number of samples from the reference distribution; RSS^*_{kb} is the residual sum of squares of the $b^t h$ sample from the reference distribution fitted in the SSMF model model using

k clusters; RSS_k is the residual sum of squares for the original data X fitted the model using the same k. The estimated gap suggests the number of prototypes/clusters ($\hat{k}$) using

$$\hat{k} = \text{smallest } k \text{ such that } \text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1},$$

where s_{k+1} is standard error that is defined as

$$s_{k+1} = sd_k \sqrt{1 + \frac{1}{B}},$$

and sd_k is the standard deviation:

$$sd_k = \sqrt{\frac{1}{B} \sum_b [\log(\text{RSS}^*_{kb}) - \frac{1}{B} \sum_b \log(\text{RSS}^*_{kb})]^2}.$$

Value

`gap` Gap value vector.

`optimal.k` The optimal number of prototypes/clusters.

`standard.error` Standard error vector.

Author(s)

Wenxuan Liu

References

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the Number of Clusters in a Data Set via the Gap Statistic. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 63(2), 411–423. <doi:10.1111/1467-9868.00293>

Examples

```
# example code

data <- SimulatedDataset

k <- 6

rss <- rep(NA, k)
for(i in 1:k){
  rss[i] <- ssmf(data = data, k = i)$SSE
}

gap(data = data, rss = rss)
```

`init` *Initialise the membership matrix H or prototype matrix W.*

Description

This function initialises the $H_{n\times k}$ matrix or the $W_{k\times p}$ matrix to start the SSMF model. This function is often used in conjunction with the function ssmf( ). Also, the code can be run separately from the function ssmf( ). This function returns to simplex-structured soft membership matrix H and prototype matrix W.

Usage

```
init(
  data = NULL,
  k = NULL,
  method = c("kmeans", "uniform", "dirichlet", "nmf")
)
```

Arguments

`data`	Data matrix or data frame.
`k`	The number of prototypes/clusters.
`method`	Character: 'kmeans', 'uniform', 'dirichlet' or 'nmf'. If there are more than one method, the default is selecting the first method in the vector.

Details

'kmeans': create the W matrix using the centres of the kmeans output; create the H matrix by converting the classification into a binary matrix.

'uniform': create the H matrix by sampling the values from uniform distribution and making the rows of the matrix lie in the unit simplex; group the observations with their maximum memberships and create the W matrix by combining the mean vector in each group.

'dirichlet': create the H matrix by sampling the values from Dirichlet distribution; group the observations with their maximum memberships and create the W matrix by combining the mean vector in each group.

'nmf': create the W matrix using the matrix of basic components from NMF model; the coefficient matrix is acquired from NMF model, then the H is created by making the rows of the coefficient matrix lie in the unit simplex.

Value

Initialised H, W matrix.

Author(s)

Wenxuan Liu

Examples

```
# example code

init(data = SimulatedDataset, k = 4, method = 'kmeans')
```

---

sARI *Soft adjusted Rand index.*

Description

Soft adjusted Rand index, a soft agreement measure for class partitions incorporating assignment probabilities.

Usage

```
sARI(partition1, partition2)
```

Arguments

partition1	Numeric matrix/data frame of the probabilities of assignment of observations in partition 1 (membership matrix).
partition2	Numeric matrix/data frame of the probabilities of assignment of observations in partition 2 (membership matrix).

Value

Soft adjusted Rand index.

Author(s)

Wenxuan Liu

References

Flynt, A., Dean, N. & Nugent, R. (2019) sARI: a soft agreement measure for class partitions incorporating assignment probabilities. Adv Data Anal Classif 13, 303–323 (2019). <doi:10.1007/s11634-018-0346-x>

SimulatedDataset *A simulated metabolomic dataset.*

Description

A simulated metabolomic data set containing 138 variables for 177 individuals.

Usage

```
data(SimulatedDataset)
```

Format

A data frame with 177 rows and 138 columns.

SimulatedMemberships *A simulated membership matrix.*

Description

A simulated membership matrix containing 4 cluster memberships for 177 individuals.

Usage

```
data(SimulatedMemberships)
```

Format

A data frame with 177 rows and 4 columns.

SimulatedPrototypes *A simulated prototype matrix.*

Description

A simulated prototype matrix containing 4 cluster prototypes.

Usage

```
data(SimulatedPrototypes)
```

Format

A data frame with 4 rows and 138 columns.

ssmf *Simplex-structured matrix factorisation algorithm (SSMF).*

Description

This function implements on SSMF on a data matrix or data frame.

Usage

```
ssmf(
  data,
  k,
  H = NULL,
  W = NULL,
  meth = c("kmeans", "uniform", "dirichlet", "nmf"),
  lr = 0.01,
  nruns = 50
)
```

Arguments

data	Data matrix or data frame.
k	The number of prototypes/clusters.
H	Matrix, user input H matrix to start the algorithm. If input is empty, the function will initialise H matrix automatically.
W	Matrix, user input W matrix to start the algorithm. If input is empty, the function will initialise W matrix automatically.
meth	Specification of method to initialise the W and H matrix, see 'method' in `init()`.
lr	Optimisation learning rate.
nruns	The maximum times of running the algorithm.

Details

Let $X \in R^{n \times p}$ be the data set with n observations and p variables. Given an integer $k \ll \min(n, p)$, the data set is clustered by simplex-structured matrix factorisation (SSMF), which aims to process soft clustering and partition the observations into k fuzzy clusters such that the sum of squares from observations to the assigned cluster prototypes is minimised. SSMF finds $H_{n \times k}$ and $W_{k \times p}$, such that

$$X \approx HW,$$

A cluster prototype refers to a vector that represent the characteristics of a particular cluster, denoted by $w_r \in \mathbb{R}^p$, where r is the r^{th} cluster. A cluster membership vector $h_i \in \mathbb{R}^k$ describes the proportion of the cluster prototypes of the i^{th} observation. W is the prototype matrix where each row is the cluster prototype and H is the soft membership matrix where each row gives the soft cluster

membership of each observation. The problem of finding the approximate matrix factorisation is solved by minising residual sum of squares (RSS), that is

$$\mathrm{RSS} = \|X - HW\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{p}\{X_{ij} - (HW)_{ij}\}^2,$$

such that $\sum_{r=1}^{k} h_{ir} = 1$ and $h_{ir} \geq 0$.

Value

`W` The optimised W matrix, containing the values of prototypes.

`H` The optimised H matrix, containing the values of soft memberships.

`SSE` The residuals sum of square.

Author(s)

Wenxuan Liu

References

Abdolali, Maryam & Gillis, Nicolas. (2020). Simplex-Structured Matrix Factorization: Sparsity-based Identifiability and Provably Correct Algorithms. <doi:10.1137/20M1354982>

Examples

```
library(MetabolSSMF)

# Initialisation by user
data <- SimulatedDataset
k <- 4

## Initialised by kmeans
fit.km <- kmeans(data, centers = k)

H <- mclust::unmap(fit.km$cluster)
W <- fit.km$centers

fit1 <- ssmf(data, k = k, H = H) #start the algorithm from H
fit2 <- ssmf(data, k = k, W = W) #start the algorithm from W

# Initialisation inside the function
fit3 <- ssmf(data, k = 4, meth = 'dirichlet')
fit4 <- ssmf(data, k = 4)
```

Index