Package: M3JF (via r-universe)

August 27, 2024

Type Package

Title Multi-Modal Matrix Joint Factorization for Integrative Multi-Omics Data Analysis

Version 0.1.0

Date 2023-07-25

Description Multi modality data matrices are factorized conjointly into the multiplication of a shared sub-matrix and multiple modality specific sub-matrices, group sparse constraint is applied to the shared sub-matrix to capture the homogeneous and heterogeneous information, respectively. Then the samples are classified by clustering the shared sub-matrix with kmeanspp(), a new version of kmeans() developed here to obtain concordant results. The package also provides the cluster number estimation by rotation cost. Moreover, cluster specific features could be retrieved using hypergeometric tests.

Imports MASS, SNFtool, dplyr, InterSIM, stats

Suggests knitr,rmarkdown

VignetteBuilder knitr

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Xiaoyao Yin [aut, cre]

Maintainer Xiaoyao Yin <xyyin@xmail.ncba.ac.cn>

Repository CRAN

Date/Publication 2023-08-14 08:20:06 UTC

Contents

`cost` *Calculate the cost defined by the objective function*

Description

Calculate the cost defined by the objective function

Usage

```
cost(WL, init_list, lambda)
```

Arguments

`WL`	a list of multiple modality data matrices
`init_list`	a list of the initialized modality specific sub-matrices list Hi and shared sub-matrix E
`lambda`	a parameter to set the relative weight of the L1,infinity norm defined on sub-matrices list E

Value

res, a real data value of the cost

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
init_list <- initialize_WL(temp_data,k=4)
```

```
update_H_list <- update_H(temp_data,init_list)
lambda <- 0.01
update_E_list <- update_E(temp_data,update_H_list,lambda)
new_cost <- cost(temp_data,update_E_list,lambda)
```

`crimmix_data_gen` *Generate the simulated dataset with three modalities with the package crimmix*

Description

Generate the simulated dataset with three modalities with the package crimmix

Usage

```
crimmix_data_gen(nclust=4, n_byClust=c(10,20,5,25),
feature_nums=c(1000,500,5000), noises=c(0.5,0.01,0.3),props=c(0.005,0.01,0.02))
```

Arguments

`nclust`	number of clusters
`n_byClust`	number of samples per cluster
`feature_nums`	number of features in each modality
`noises`	percentage of noise adding to each modality
`props`	proportion of cluster related features in each modality

Value

res, a list of length 2, where the first element is a list of simulated data, while the second element is a vector indicating the true label of each sample

Examples

```
crimmix_data <- crimmix_data_gen(nclust=4, n_byClust=c(10,20,5,25),
feature_nums=c(1000,500,5000), noises=c(0.5,0.01,0.3),props=c(0.005,0.01,0.02))
```

`feature_screen_sd` *Screen the cluster related features via hypergeometric test p value and distribution standard derivation*

Description

Screen the cluster related features via hypergeometric test p value and distribution standard derivation

Usage

```
feature_screen_sd(feature_list, sig_num = 20)
```

Arguments

`feature_list`	a data list, which is the output of feature_selection function
`sig_num`	the number of significant features for each cluster

Value

selected_features, a list the same long as the cluster number, each element is a sub-list with two vectors, one for the over-expressed features, one for the under-expressed features for the current cluster

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
M3JF_res <- M3JF(temp_data,k=4)
feature_list <- feature_selection(temp_data[[1]],M3JF_res$cluster_res,z_score=TRUE,
upper_bound=1, lower_bound=-1)
selected_features <- feature_screen_sd(feature_list,sig_num=20)
```

`feature_selection` *Select the cluster related features via hypergeometric test*

Description

Select the cluster related features via hypergeometric test

Usage

```
feature_selection(
  X,
  clusters,
  z_score = FALSE,
  upper_bound,
  lower_bound,
  p.adjust.method = "BH"
)
```

Arguments

`X`	the feature matrix to be analyzed, with rows as samples and columns as features
`clusters`	the numeric cluster results with number specifying the cluster
`z_score`	a binary value to specify whether to calculate z-score for X first
`upper_bound`	values larger than this value should be treated as over-expressed
`lower_bound`	values smaller than this value should be treated as under-expressed
`p.adjust.method`	the p value adjust method, defalut as 'BH'

Value

results, a list, which is as long as (cluster number+2), with the first (cluster number) element as two sub-list, each composing a feature vector and a FDR vector. The last two elements are two matrices, one is the matrix representing the fraction of over-express samples in each cluster for each features , and the other represents that of under-express.

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
M3JF_res <- M3JF(temp_data,k=4)
```

```
feature_list <- feature_selection(temp_data[[1]],M3JF_res$cluster_res,z_score=TRUE,
upper_bound=1, lower_bound=-1)
```

`initialize_WL` *Initialize the shared sub-matrix E and modality specific sub-matrices list Hi*

Description

Initialize the shared sub-matrix E and modality specific sub-matrices list Hi

Usage

```
initialize_WL(WL, k)
```

Arguments

`WL`	a list of multiple modality data matrices
`k`	the cluster number

Value

res, a list of length N+3, where N is the number of data modality. the first N elements are the modality specific sub-matrices list Hi, the (N+1) element is the shared sub-matrix E, the last two elements are the loss defined on the shared sub-matrix E and modality specific sub-matrices list Hi.

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
init_list <- initialize_WL(temp_data,k=4)
```

iNMF_data_gen — *Generate the simulated dataset with three modalities as the work iNMF*

Description

Generate the simulated dataset with three modalities as the work iNMF

Usage

```
iNMF_data_gen(Xs_dim_list=list(c(100,100),c(100,100),c(100,100)),
mod_dim_list=list(matrix(c(20,30,20,30,20,30,20,30),4,2),
matrix(c(20,20,30,30,20,30,20,30),4,2),
matrix(c(26,24,26,24,20,30,20,30),4,2)),e_u=0.15, e_s=0.9, e_h=0)
```

Arguments

`Xs_dim_list`	a list of data matrix dimensions for multiple modality data
`mod_dim_list`	a list of the dimensions of each cluster and their features
`e_u`	the level of uniform noise
`e_s`	signal to noise ratio
`e_h`	block adding probability

Value

res, a list of length 2, where the first element is a list of simulated data, while the second element is a vector indicating the true label of each sample.

Examples

```
iNMF_data <- iNMF_data_gen(Xs_dim_list=list(c(100,100),c(100,100),c(100,100)),
mod_dim_list=list(matrix(c(20,30,20,30,20,30,20,30),4,2),
matrix(c(20,20,30,30,20,30,20,30),4,2),
matrix(c(26,24,26,24,20,30,20,30),4,2)),e_u=0.15, e_s=0.9, e_h=0)
```

intersim_data_gen — *Generate the simulated dataset with three modalities with the package InterSIM*

Description

Generate the simulated dataset with three modalities with the package InterSIM

Usage

```
intersim_data_gen(prop=c(0.20,0.30,0.27,0.23), n_sample=500)
```

Arguments

prop	proportion of samples for each cluster
n_sample	the number of samples

Value

res, a list of length 2, where the first element is a list of simulated data, while the second element is a vector indicating the true label of each sample.

Examples

```
library(InterSIM)
intersim_data <- intersim_data_gen(prop=c(0.20,0.30,0.27,0.23), n_sample=500)
```

---

kmeanspp *A new version of kmeans that generates stable cluster result*

Description

A new version of kmeans that generates stable cluster result

Usage

```
kmeanspp(X, k)
```

Arguments

X	a data matrix with each row as a sample and each column as a feature
k	the cluster number

Value

res, the cluster result generated by this function

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
init_list <- initialize_WL(temp_data,k=4)
lambda <- 0.01
update_E_list <- update_E(temp_data,init_list,lambda)
cluster_res <- kmeanspp(update_E_list[[4]],4)
```

M3JF *Multi-Modal Matrix Joint Factorization*

Description

Multi-Modal Matrix Joint Factorization

Usage

```
M3JF(WL, lambda = 0.01, theta = 10^-6, k)
```

Arguments

`WL`	a list of multiple modality data matrices
`lambda`	the parameter to set the relative weight of the group sparse constraint
`theta`	threshold for the stopping criteria
`k`	cluster number

Value

result, a list of 3 elements, the first element is a list comprising the shared sub-matrix and the modality specific sub-matrices. The second element is a vector of the clustering result. The third element is a vector of the cost in each step during optimization.

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
M3JF_res <- M3JF(temp_data,k=4)
```

RotationCostBestGivenGraph *Evaluate the cluster number of multiple modality data*

Description

Evaluate the cluster number of multiple modality data

Usage

```
RotationCostBestGivenGraph(W, NUMC = 2:5)
```

Arguments

W	a list of multiple modality data matrices
NUMC	a vector specify the data range to select best cluster number

Value

quality, a vector of rotation cost the same long as NUMC, where each element is the rotation cost value of the corresponding cluster number.

Examples

```
library(InterSIM)
library(SNFtool)
sim.data <- InterSIM(n.sample=100, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
dat <- lapply(temp_data, function(dd) {
  dd <- as.matrix(dd)
  dd1 <- dist2(dd,dd)
  W1 <- affinityMatrix(dd1, K = 10, sigma = 0.5)
})
W <- SNF(dat, 10, 10)
clu_eval <- RotationCostBestGivenGraph(W,2:10)
```

---

simulateY *Generate the simulated dataset with specified parameters*

Description

Generate the simulated dataset with specified parameters

Usage

```
simulateY(nclust = 4, n_byClust = c(10,20,5,25), J=1000, prop = 0.01,
noise = 0.1,flavor =c("normal", "beta", "binary"),
params = list(c(mean = 1,sd = 1)))
```

Arguments

`nclust`	number of clusters
`n_byClust`	number of samples per cluster
`J`	number of features in each modality
`prop`	proportion of cluster related features
`noise`	percentage of noise adding to each modality
`flavor`	a vector indicating the data type
`params`	a list indicating the mean and standard derivation of the simulated data

Value

res, a list of length 2, where the first element is a list of simulated data, while the second element is a vector indicating the true label of each sample

Examples

```
temp_data <- simulateY(nclust = 4, n_byClust = c(10,20,5,25), J=1000,
prop = 0.01, noise = 0.1,flavor =c("normal", "beta", "binary"),
params = list(c(mean = 1,sd = 1)))
```

`update_E` *Update sub-matrix E*

Description

Update sub-matrix E

Usage

```
update_E(WL, init_list, lambda)
```

Arguments

`WL`	a list of multiple modality data matrices
`init_list`	a list of the initialized modality specific sub-matrices list Hi and shared sub-matrix E
`lambda`	a parameter to set the relative weight of the L1,infinity norm defined on sub-matrices list E

Value

update_E_list, the data list init_list with the shared sub-matrix E updated.

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
init_list <- initialize_WL(temp_data,k=4)
update_H_list <- update_H(temp_data,init_list)
lambda <- 0.01
update_E_list <- update_E(temp_data,update_H_list,lambda)
```

`update_H` *Update sub-matrices list Hi*

Description

Update sub-matrices list Hi

Usage

```
update_H(WL, init_list)
```

Arguments

`WL`	a list of multiple modality data matrices
`init_list`	a list of the initialized modality specific sub-matrices list Hi and shared sub-matrix E

Value

update_H_list, the data list init_list with the modality specific sub-matrices list Hi updated.

Examples

```
library(InterSIM)
sim.data <- InterSIM(n.sample=500, cluster.sample.prop = c(0.20,0.30,0.27,0.23),
delta.methyl=5, delta.expr=5, delta.protein=5,p.DMP=0.2, p.DEG=NULL,
p.DEP=NULL,sigma.methyl=NULL, sigma.expr=NULL, sigma.protein=NULL,cor.methyl.expr=NULL,
cor.expr.protein=NULL,do.plot=FALSE, sample.cluster=TRUE, feature.cluster=TRUE)
sim.methyl <- sim.data$dat.methyl
sim.expr <- sim.data$dat.expr
sim.protein <- sim.data$dat.protein
temp_data <- list(sim.methyl, sim.expr, sim.protein)
init_list <- initialize_WL(temp_data,k=4)
update_H_list <- update_H(temp_data,init_list)
```

Index