Package 'OSNMTF'

October 12, 2022

Type Package

Title Orthogonal Sparse Non-Negative Matrix Tri-Factorization

Version 0.1.0

Author Xiaoyao Yin

Maintainer Xiaoyao Yin <yinxy1992@sina.com>

Description A novel method to implement cancer subtyping and subtype specific drug targets identification via non-negative matrix tri-factorization.
To improve the interpretability, we introduce orthogonal constraint to the row coefficient matrix and column coefficient matrix. To meet the prior knowledge
that each subtype should be strongly associated with few gene sets, we introduce sparsity constraint to the association sub-matrix. The average residue was
introduced to evaluate the row and column cluster numbers. This is part of the work ``Liver Cancer Analysis via Orthogonal Sparse Non-Negative Matrix Tri-
Factorization" which will be submitted to BBRC.

Imports dplyr, MASS, stats

Depends R (>= 3.4.4)

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-11-28 13:50:02 UTC

R topics documented:

`affinityMatrix` *Calculate the similarity matrix*

Description

To calculate the similarity matrix with the same method in package M2SMF, for asymmetric case

Usage

```
affinityMatrix(Diff, K = 20, sigma = 0.5)
```

Arguments

`Diff`	The distance matrix to culculate the similarity
`K`	The number of neighbours to culculate the similarity
`sigma`	A hyper-parameter to culculate the similarity

Value

The similarity matrix

Author(s)

Xiaoyao Yin

Examples

```
data1 <- matrix(0,100,100)
data2 <- matrix(0,80,100)
for (i in 1:20)
{
  data1[i,] <- rnorm(100,10,1)
}
for (i in 21:40)
```

```
{
  data1[i,] <- rnorm(100,20,1)
}
for (i in 41:60)
{
  data1[i,] <- rnorm(100,30,1)
}
for (i in 61:80)
{
  data1[i,] <- rnorm(100,40,1)
}
for (i in 81:100)
{
  data1[i,] <- rnorm(100,50,1)
}
for (i in 1:20)
{
  data2[i,] <- rnorm(100,5,1)
}
for (i in 21:40)
{
  data2[i,] <- rnorm(100,10,1)
}
for (i in 41:60)
{
  data2[i,] <- rnorm(100,15,1)
}
for (i in 61:80)
{
  data2[i,] <- rnorm(100,20,1)
}
new_data1 <- Standard_Normalization(data1)
new_data2 <- Standard_Normalization(data2)
Diff <- dist2eu(new_data1,new_data2)
simi_matr1 <- affinityMatrix(Diff, K = 20, sigma = 0.5)
```

ASR *Average Residue*

Description

To calculate average residues of the bi-clustering results

Usage

```
ASR(row_cluster,col_cluster,W)
```

Arguments

row_cluster	The cluster results of the rows of W, this value should be a vector whose length is the same as the number of rows in W
col_cluster	The cluster results of the columns of W, this value should be a vector whose length is the same as the number of columns in W
W	The matrix to be factorized

Value

The average residues of the bi-clustering results

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
OSNMTF_res <- OSNMTF(W,k=5,l=4)
row_cluster <- OSNMTF_res[[2]][[1]]
column_cluster <- OSNMTF_res[[2]][[2]]
ASR_value <- ASR(row_cluster,column_cluster,W)
```

---

cost *Calculate the cost*

Description

A function to calculate the cost of the objective function

Usage

```
cost(W,init_list,lambda=0.2)
```

Arguments

W	The matrix to be factorized
init_list	A list containing the updated results in this iteration
lambda	A parameter to set the relative weight of the sparsity constraint

Value

A number indicating the total cost of the objective function

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
init_list <- initialization(W,k=5,l=4)
update_L_list <- update_L(W,init_list)
update_B_list <- update_B(W,update_L_list)
update_R_list <- update_R(W,update_B_list)
update_C_list <- update_C(W,update_R_list,lambda=0.2,rho=1.1)
temp_cost <- cost(W,init_list,lambda=0.2)
```

---

dist2eu *Euclidean Distance*

Description

The distance matrix of the two group of samples

Usage

```
dist2eu(X,C)
```

Arguments

X	The first samples matrix
C	The second samples matrix

Value

The distance matrix

Author(s)

Xiaoyao Yin

Examples

```
data1 <- matrix(0,100,100)
data2 <- matrix(0,80,100)
for (i in 1:20)
{
  data1[i,] <- rnorm(100,10,1)
}
for (i in 21:40)
{
  data1[i,] <- rnorm(100,20,1)
}
for (i in 41:60)
{
```

```
  data1[i,] <- rnorm(100,30,1)
}
for (i in 61:80)
{
  data1[i,] <- rnorm(100,40,1)
}
for (i in 81:100)
{
  data1[i,] <- rnorm(100,50,1)
}
for (i in 1:20)
{
  data2[i,] <- rnorm(100,5,1)
}
for (i in 21:40)
{
  data2[i,] <- rnorm(100,10,1)
}
for (i in 41:60)
{
  data2[i,] <- rnorm(100,15,1)
}
for (i in 61:80)
{
  data2[i,] <- rnorm(100,20,1)
}
new_data1 <- Standard_Normalization(data1)
new_data2 <- Standard_Normalization(data2)
dist1 <- dist2eu(new_data1,new_data2)
```

`initialization` *initialize the values used in NMTFOSC*

Description

initialize the values which will be updated in NMTFOSC

Usage

```
initialization(W,k,l)
```

Arguments

`W`	The matrix to be factorized
`k`	A parameter to specify the row cluster number
`l`	A parameter to specify the column cluster number

Value

A list with 6 elements, corresponding to the matrices L,C,R,B,Y and the penalty parameter miu

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
init_list <- initialization(W,k=5,l=4)
```

MSR *Mean Residue*

Description

To calculate mean residue of a sub-matrix block of W, indexed by a row cluster and a column cluster

Usage

```
MSR(Block)
```

Arguments

Block	The sub-matrix block of W, indexed by a row cluster and a column cluster

Value

The mean residue of the block

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
OSNMTF_res <- OSNMTF(W,k=5,l=4)
row_cluster <- OSNMTF_res[[2]][[1]]
column_cluster <- OSNMTF_res[[2]][[2]]
temp_rows <- which(row_cluster==1,TRUE)
temp_cols <- which(column_cluster==1,TRUE)
MSR_value <- MSR(W[temp_rows,temp_cols])
```

OSNMTF *The algorithm OSNMTF*

Description

Factorize matrix W into the multiplication of L, C and R, with L and R being orthogonal and C being sparse. Then the row cluster results and column cluster results are obtained from L and R.

Usage

```
OSNMTF(W,lambda=0.2,theta=10^-4,k,l)
```

Arguments

W	The matrix to be factorized
lambda	A parameter to set the relative weight of the sparsity constraints
theta	A parameter to determine the convergence
k	A parameter to specify the row cluster number
l	A parameter to specify the column cluster number

Value

A list containing the clustering result

sub_matrices	a list containing the matrix L, C, R
cluster_results	a list containing the row cluster results and the column cluster results

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
OSNMTF_res <- OSNMTF(W,k=5,l=4)
```

simu_data_generation *Generate simulation data*

Description

To generate the simulation data matrix

Usage

```
simu_data_generation()
```

Value

The simulated data matrix

Author(s)

Xiaoyao Yin

Examples

```
simu_data <- simu_data_generation()
```

Standard_Normalization *Standard Normalization*

Description

To normalize the data matrix by column

Usage

```
Standard_Normalization(x)
```

Arguments

x	The data matrix to be normalized

Value

The normalized matrix

Author(s)

Xiaoyao Yin

Examples

```
data1 <- matrix(0,100,100)
data2 <- matrix(0,80,100)
for (i in 1:20)
{
  data1[i,] <- rnorm(100,10,1)
}
for (i in 21:40)
{
  data1[i,] <- rnorm(100,20,1)
}
for (i in 41:60)
{
  data1[i,] <- rnorm(100,30,1)
}
for (i in 61:80)
{
  data1[i,] <- rnorm(100,40,1)
}
for (i in 81:100)
{
  data1[i,] <- rnorm(100,50,1)
}
new_data1 <- Standard_Normalization(data1)
```

---

update_B *Update sub-matrix B*

Description

Update sub-matrix B

Usage

```
update_B(W,update_L_list)
```

Arguments

W	The matrix to be factorized
update_L_list	A list containing the updated results in this iteration after running the function update_L

Value

A list the same as update_L_list with the matrix B updated

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
init_list <- initialization(W,k=5,l=4)
update_L_list <- update_L(W,init_list)
update_B_list <- update_B(W,update_L_list)
```

---

update_C *Update sub-matrix C*

Description

Update sub-matrix C

Usage

```
update_C(W,update_R_list,lambda=0.2,rho=1.1)
```

Arguments

W	The matrix to be factorized
update_R_list	A list containing the updated results in this iteration after running the function update_R
lambda	A parameter to set the relative weight of the sparsity constraints
rho	A parameter used in the augmented lagrange multiplier method

Value

A list the same as update_R_list with the matrix C updated

Author(s)

Xiaoyao Yin

Examples

```
W <- simu_data_generation()
init_list <- initialization(W,k=5,l=4)
update_L_list <- update_L(W,init_list)
update_B_list <- update_B(W,update_L_list)
update_R_list <- update_R(W,update_B_list)
update_C_list <- update_C(W,update_R_list,lambda=0.2,rho=1.1)
```

update_L *Update sub-matrix L*

Description

Update sub-matrix L

Usage

```
update_L(W,init_list)
```

Arguments

W	The matrix to be factorized
init_list	A list containing the updated results in this iteration

Value

A list the same as init_list with the matrix L updated

Examples

```
W <- simu_data_generation()
init_list <- initialization(W,k=5,l=4)
update_L_list <- update_L(W,init_list)
```

update_R *Update sub-matrix R*

Description

Update sub-matrix R

Usage

```
update_R(W,update_B_list)
```

Arguments

W	The matrix to be factorized
update_B_list	A list containing the updated results in this iteration after running the function update_B

Value

A list the same as update_B_list with the matrix R updated

Examples

```
W <- simu_data_generation()
init_list <- initialization(W,k=5,l=4)
update_L_list <- update_L(W,init_list)
update_B_list <- update_B(W,update_L_list)
update_R_list <- update_R(W,update_B_list)
```

Index