Package 'TDbasedUFE'

May 15, 2024

Type Package

Title Tensor Decomposition Based Unsupervised Feature Extraction

Version 1.4.0

Description This is a comprehensive package to perform
Tensor decomposition based unsupervised feature extraction.
It can perform unsupervised feature extraction.
It uses tensor decomposition.
It is applicable to gene expression, DNA methylation, and
histone modification etc.
It can perform multiomics analysis.
It is also potentially applicable to single cell omics data sets.

biocViews GeneExpression, FeatureExtraction, MethylationArray, SingleCell

License GPL-3

Encoding UTF-8

LazyData false

URL https://github.com/tagtag/TDbasedUFE

BugReports https://github.com/tagtag/TDbasedUFE/issues

Imports GenomicRanges, rTensor, readr, methods, MOFAdata, tximport, tximportData, graphics, stats, utils, shiny

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/TDbasedUFE

git_branch RELEASE_3_19

git_last_commit 63e4439

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-14

Author Y-h. Taguchi [aut, cre] (<https://orcid.org/0000-0003-0867-8986>)

Maintainer Y-h. Taguchi <tag@granular.com>

Contents

`TDbasedUFE-package` *TDbasedUFE: Tensor Decomposition Based Unsupervised Feature Extraction*

Description

This is a comprehensive package to perform Tensor decomposition based unsupervised feature extraction. It can perform unsupervised feature extraction. It uses tensor decomposition. It is applicable to gene expression, DNA methylation, and histone modification etc. It can perform multiomics analysis. It is also potentially applicable to single cell omics data sets.

Author(s)

Maintainer: Y-h. Taguchi <tag@granular.com> (ORCID)

See Also

Useful links:

- https://github.com/tagtag/TDbasedUFE
- Report bugs at https://github.com/tagtag/TDbasedUFE/issues

computeHosvd *Title Compute higher order singular value decomposition*

Description

Title Compute higher order singular value decomposition

Usage

```
computeHosvd(Z, dims = c(10, dim(attr(Z, "value"))[-1]), scale = TRUE)
```

Arguments

Z	array that includes omics data
dims	dimensions to be computed by HOSVD
scale	If value is scaled

Value

List that includes output from HOSVD

Examples

```
Z <- PrepareSummarizedExperimentTensor(
sample=matrix(as.character(seq_len(6)),c(3,2)),
feature=as.character(seq_len(10)),
value=array(runif(10*3*2),c(10,3,2)))
HOSVD <- computeHosvd(Z)
```

computeHosvdSqure *Title Compute higher order singular value decomposition from the tensor generated from squared matrix*

Description

Title Compute higher order singular value decomposition from the tensor generated from squared matrix

Usage

```
computeHosvdSqure(
  Z,
  dims = unlist(lapply(dim(attr(Z, "value")), function(x) {
     min(10, x)
 })),
  scale = TRUE
)
```

Arguments

Z	A tensor including sample names, feature values, associated with featureRange and sample properties
dims	dimensions to be computed by HOSVD
scale	If value is scaled

Value

List that includes output from HOSVD

Examples

```
omics1 <- matrix(runif(100),10)
dimnames(omics1) <- list(seq_len(10),seq_len(10))
omics2 <- matrix(runif(100),10)
dimnames(omics2) <- dimnames(omics1)
Multi <- list(omics1,omics2)
Z <- PrepareSummarizedExperimentTensorSquare(
            sample=matrix(colnames(omics1),1),
            feature=list(omics1=rownames(omics1),
            omics2=rownames(omics2)),
            value=convertSquare(Multi),
            sampleData=list(NA))
            HOSVD <- computeHosvdSqure(Z)
```

---

convertSquare *Generate squared tensor from multiomics data*

Description

Generate squared tensor from multiomics data

Usage

```
convertSquare(Multi)
```

Arguments

Multi	A list that include multiomics data

Value

A tensor computed from multiomics data

Examples

```
omics1 <- matrix(runif(100),10)
dimnames(omics1) <- list(seq_len(10),seq_len(10))
omics2 <- matrix(runif(100),10)
dimnames(omics2) <- dimnames(omics1)
Multi <- list(omics1,omics2)
Z <- convertSquare(Multi)
```

`PrepareSummarizedExperimentTensor` *Title Generate feature values formatted as a tensor format*

Description

Title Generate feature values formatted as a tensor format

Usage

```
PrepareSummarizedExperimentTensor(
  sample,
  feature,
  value,
  featureRange = GRanges(NULL),
  sampleData = list(NULL)
)
```

Arguments

`sample`	Sample names
`feature`	Feature id names
`value`	Feature values
`featureRange`	Genomic coordinate attributed to feature id (if any)
`sampleData`	Sample property (labels etc)

Value

A tensor including sample names, feature id, feature values, associated with featureRange and sample properties

Examples

```
require(GenomicRanges)
Z <- PrepareSummarizedExperimentTensor(
sample=matrix(as.character(seq_len(6)),c(3,2)),
feature=as.character(seq_len(10)),
 value=array(runif(10*3*2),c(10,3,2)))
```

PrepareSummarizedExperimentTensorSquare *Title Generate feature values formatted as a tensor format from Squared matrix*

Description

Title Generate feature values formatted as a tensor format from Squared matrix

Usage

```
PrepareSummarizedExperimentTensorSquare(
  sample = list(NULL),
  feature,
  value,
  featureRange = GRanges(NULL),
  sampleData = list(NULL)
)
```

Arguments

sample	Sample names
feature	Feature id names
value	Squared Feature values
featureRange	Genomic coordinate attributed to feature id (if any)
sampleData	Sample property (labels etc)

Value

A tensor including sample names, feature values, associated with featureRange and sample properties

Examples

```
omics1 <- matrix(runif(100),10)
dimnames(omics1) <- list(seq_len(10),seq_len(10))
omics2 <- matrix(runif(100),10)
dimnames(omics2) <- dimnames(omics1)
Multi <- list(omics1,omics2)
Z <- PrepareSummarizedExperimentTensorSquare(
   sample=matrix(colnames(omics1),1),
   feature=list(omics1=rownames(omics1),
   omics2=rownames(omics2)),
   value=convertSquare(Multi),
   sampleData=list(NA))
```

`selectFeature` *Title Select features*

Description

Title Select features

Usage

```
selectFeature(HOSVD, input_all, de = 1e-04, p0 = 0.01, breaks = 100)
```

Arguments

`HOSVD`	output from HOSVD
`input_all`	Selected singular value IDs
`de`	Initial value for optimization of standard deviation
`p0`	Threshold P-value
`breaks`	The number of bins

Value

List that includes selected features and computed P-value

Examples

```
set.seed(2)
require(rTensor)
HOSVD <- hosvd(as.tensor(array(runif(10000*3*3),c(10000,3,3))),c(10,3,3))
input_all <- c(2,2)
index <- selectFeature(HOSVD,input_all,de=0.01,p0=0.01)
```

`selectFeatureSquare` *Title Select features (for tensor generated from squared matrix)*

Description

Title Select features (for tensor generated from squared matrix)

Usage

```
selectFeatureSquare(
  HOSVD,
  input_all,
  Multi,
  de = rep(1e-04, dim(HOSVD$U[[3]])[2]),
  p0 = 0.01,
  breaks = 100,
  interact = TRUE
)
```

Arguments

HOSVD	output from HOSVD applied to tensor generated from squared matrix
input_all	Selected singular value vector IDs
Multi	Multiomics data
de	Initial value for optimization of standard deviation
p0	Threshold P-value
breaks	The number of bins
interact	if interact mode or not

Value

List that includes selected features and computed P-value

Examples

```
omics1 <- matrix(runif(100000),ncol=10)
dimnames(omics1) <- list(seq_len(10000),seq_len(10))
omics2 <- matrix(runif(100000),ncol=10)
dimnames(omics2) <- dimnames(omics1)
Multi <- list(omics1,omics2)
Z <- PrepareSummarizedExperimentTensorSquare(
sample=matrix(colnames(omics1),1),
feature=list(omics1=rownames(omics1),
omics2=rownames(omics2)),
value=convertSquare(Multi),
sampleData=list(NA))
HOSVD <- computeHosvdSqure(Z)
cond <- list(0,rep(seq_len(2),each=5),c("A","B"))
input_all <- selectSingularValueVectorLarge(HOSVD,cond,input_all=c(1,1))
index <- selectFeatureSquare(HOSVD,input_all,Multi,de=c(0.1,0.1),
interact=FALSE)
```

selectSingularValueVectorLarge

Title Select singular value vectors from HOSVD (boxplot version)

Description

Title Select singular value vectors from HOSVD (boxplot version)

Usage

```
selectSingularValueVectorLarge(HOSVD, cond, input_all = NULL)
```

Arguments

HOSVD	output from HOSVD
cond	Labels to select singular value vector number
input_all	if list is not null, no interactive mode is activated but provided values are used.

Value

Selected singular value vector IDs

Examples

```
Z <- PrepareSummarizedExperimentTensor(
sample=matrix(as.character(seq_len(6)),c(3,2)),
feature=as.character(seq_len(10)),
 value=array(runif(10*3*2),c(10,3,2)))
HOSVD <- computeHosvd(Z)
 cond <- list(0,c("A","B","C"),c("A","B"))
 input_all <- selectSingularValueVectorLarge(HOSVD,cond,input_all=c(1,1))
```

selectSingularValueVectorSmall

Title Select singular value vectors from HOSVD

Description

Title Select singular value vectors from HOSVD

Usage

```
selectSingularValueVectorSmall(HOSVD, input_all = NULL)
```

Arguments

HOSVD	output from HOSVD
input_all	if ist is no null, no interactive mode is activated but provided values are used.

Value

Selected singular value vector IDs

Examples

```
Z <- PrepareSummarizedExperimentTensor(
sample=matrix(as.character(seq_len(6)),c(3,2)),
feature=as.character(seq_len(10)),
 value=array(runif(10*3*2),c(10,3,2)))
 HOSVD <- computeHosvd(Z)
input_all <- selectSingularValueVectorSmall(HOSVD,input_all=c(1,1))
```

---

tableFeatures *Title Show selected features as Table*

Description

Title Show selected features as Table

Usage

```
tableFeatures(Z, index)
```

Arguments

Z	Tensor of features
index	List that includes selected features and P-values

Value

Table list of selected features

Examples

```
set.seed(2)
require(rTensor)
HOSVD <- hosvd(as.tensor(array(runif(10000*3*3),c(10000,3,3))),c(10,3,3))
input_all <- c(2,2)
index <- selectFeature(HOSVD,input_all,de=0.01,p0=0.01)
index$index[seq_len(100)] <- TRUE
Z <- PrepareSummarizedExperimentTensor(
sample=matrix(as.character(seq_len(9)),c(3,3)),
        feature=as.character(seq_len(10000)),
        value=array(runif(10000*3*3),c(10,3,3)))
head(tableFeatures(Z,index))
```

`tableFeaturesSquare` *Title Show selected features as Table (for Squared one)*

Description

Title Show selected features as Table (for Squared one)

Usage

```
tableFeaturesSquare(Z, index, id)
```

Arguments

`Z`	Tensor of features
`index`	List that includes selected features and P-values
`id`	feature to be shown

Value

Table list of selected features

Examples

```
omics1 <- matrix(runif(100000),ncol=10)
dimnames(omics1) <- list(seq_len(10000),seq_len(10))
omics2 <- matrix(runif(100000),ncol=10)
dimnames(omics2) <- dimnames(omics1)
Multi <- list(omics1,omics2)
Z <- PrepareSummarizedExperimentTensorSquare(
sample=matrix(colnames(omics1),1),
feature=list(omics1=rownames(omics1),
omics2=rownames(omics2)),
value=convertSquare(Multi),
sampleData=list(NA))
HOSVD <- computeHosvdSqure(Z)
cond <- list(0,rep(seq_len(2),each=5),c("A","B"))
input_all <- selectSingularValueVectorLarge(HOSVD,cond,input_all=c(1,1))
index <- selectFeatureSquare(HOSVD,input_all,Multi,de=c(0.1,0.1),
interact=FALSE)
index[[1]]$index[1:100]<-TRUE
index[[1]]$p.value[1:100] <- 1e-3
tableFeaturesSquare(Z,index,1)
```

Index