Package ‘tensorApp’

February 16, 2020

Type Package

Title tensorApp

Version 0.1.0

Author Xu Liu [aut,cre].

Maintainer Xu Liu <liu.xu@sufe.edu.cn>

Description
High-order SVD approximation of a tensor by Tucker or CP decomposition and rank selection.

License GPL (>= 2)

Imports Rcpp (>= 0.11.15), RcppEigen (>= 0.3.2.3.0)

LinkingTo Rcpp, RcppEigen

RoxygenNote 6.0.1

NeedsCompilation yes

Repository github

URL https://github.com/xliusufe/tensorApp

Encoding UTF-8

R topics documented:

tensorApp-package	*High-order SVD approximation of a tensor* `Y` *by Tucker or CP decomposition and selection of ranks*

Description

High-order SVD approximation of a tensor `Y` by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition and selection of ranks. Alternating Least Squares algorithm is applied to Tuchker decomposition, and both Alternating Least Squares algorithm or Tensor Power Method are applied to CP decomposition. This package provides several generator functions, which generate low-rank tensor or low-rank semi-symmetric tensor.

Details

High-order SVD approximation of a tensor `Y` by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition and selection of ranks.

Author(s)

Xu Liu

Maintainer: Xu Liu <liu.xu@sufe.edu.cn>

References

Allen, G., (2012). Sparse higher-order principal components analysis, in: International Conference on Artificial Intelligence and Statistics, pp. 27-36.

De Lathauwer, L., De Moor, B., Vandewalle, J., (2000). A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications, **21**, 1253-1278.

De Lathauwer, L., De Moor, B., Vandewalle, J., (2000). On the best rank-1 and rank-$(r_1, r_2, \cdots, r_n)$ approximation of higher-order tensors. SIAM Journal on Matrix Analysis and Applications, **21**, 1324-1342.

Kolda, T.G., (2001). Orthogonal tensor decompositions. SIAM Journal on Matrix Analysis and Applications, **23**, 243-255.

Kolda, T., Bader, B., (2009). Tensor decompositions and applications. SIAM Review, **51**, 455-500.

cpals	*High-order SVD approximation of a tensor* `Y` *by CP decomposition*

Description

High-order SVD approximation of a tensor `Y` by CANDECOMP/PARAFAC (CP) decomposition with preset rank or rank to be selected. The Alternating Least Squares (als) algorithm is applied.

Usage

```
cpals(Y=NULL, d0=NULL, dims=NULL, dr=10, isfixr=FALSE,
                              D0=NULL, eps=1e-4, max_step=50, thresh=1e-6)
```

Arguments

- `Y` An array with dimension `dims`, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode `d0`-unfolding of tensor in $\mathcal{R}^{n_1\times\cdots\times n_d}$, where $N = n_1\times\cdots\times n_d$.
- `d0` `d0` is the mode. `Y` is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if `Y` is an array with dimension `dims`.
- `dims` The size of tensor `Y`, which is a d-vector $(n_1, \cdots, n_d)$. `dims` can be NULL (the default) if `Y` is an array with dimension `dims`. If the length of `dims` is 2, it is the ordinary SVD decomposition of a matrix.
- `dr` The user-specified rank for CP decomposition. Default is `10`.
- `isfixr` A logical value indicating whether the rank is fixed. The rank is selected automatically if it is `FALSE`. Default is `FALSE`.
- `D0` A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, `D0=list(`$U_1 = U_1, \cdots, U_d = U_d, S = S$`)`. By default, initial matrices are provided by random.
- `eps` Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than `eps`. Default is `1e-4`.
- `max_step` Maximum number of iterations. Default is 50.
- `thresh` Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than `eps`. Default is `1e-6`.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-`d0` unfolding, and approximates `Y`.

Value

- `Tnew` Approximation of `Y`.
- `Tn` A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, `Tn=list(`$U_1 = U_1, \cdots, U_d = U_d, S = S$`)`.
- `ranks` The ranks of estimated tensor `Tnew`. It is an integer.

See Also

hosvd_dr

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
```

```
Y <- U%*%S0%*%t(tmp)

fit <- cpals(Y,N,dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)
```

---

cpsym2 *High-order SVD approximation of a tensor* Y *by CP decomposition*

Description

High-order SVD approximation of a semi-symmetric tensor Y by CANDECOMP/PARAFAC (CP) decomposition with preset rank or rank to be selected. The Tensor Power Method is applied. The semi-symmetric tensor means that both the mode-r1 and mode-r2 unfoldings are equal. For semi-symmetric tensor approximation, the r1 and r2 dimensions must be small than others.

Usage

```
cpsym2(Y=NULL, r1=1, r2=2, d0=NULL, dims=NULL, dr=10, D0=NULL, isfixr=FALSE,
                              isOrth=FALSE, eps=1e-4, max_step=50, thresh=1e-6)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1\times\cdots\times n_d}$, where $N = n_1\times\cdots\times n_d$.
r1	Both r1 and r2 are the user-specified modes, which means that both the mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
r2	Both r1 and r2 are the user-specified modes, which means that both the mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
dr	The user-specified rank for CP decomposition. Default is 10.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list($U_1 = U_1, \cdots, U_d = U_d, S = S$). By default, initial matrices are provided by random.
isfixr	A logical value indicating whether the rank is fixed. The rank is selected automatically if it is FALSE. Default is FALSE.
isOrth	A logical value indicating whether it outputs orthonognal PCs if CP decomposition is used. Default is FALSE.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-4.
max_step	Maximum number of iterations. Default is 50.
thresh	Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew. It is an integer.

See Also

hosvd_dr

Examples

```
dims <- c(6,6,8,7,7)
N <- length(dims)
ranks <- rep(2,N)
dm <- prod(ranks)
S1 <- matrix(runif(dm,3,7),nrow = ranks[1])
S2 <- ttu(S1,1,2,ranks)
S1 <- (S1+S2)/2
S0 <- ttu(S1,1,N,ranks)

T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
Uj <- kronecker(tmp,tmp)
for(k in 3:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- qr.Q(qr(T1))
  Uj <- kronecker(tmp,Uj)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(Uj)

fit <- cpsym2(Y,r1=1,r2=2,d0=N,dims=dims)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
U1 <- fit$Tn[[r1]]
U2 <- fit$Tn[[r2]]
TNew1 <- ttu(Tnew,N,r1,dims)
TNew2 <- ttu(Tnew,N,r2,dims)
```

---

gtcp	*Generate a low-rank tensor characterizing the form of CP decomposition*

Description

This function generates a low-rank tensor characterizing the form of CP decomposition with dimension dims and factors lambda.

Usage

```
gtcp(dims, lambda=NULL, d0=NULL, dr=NULL, seed_id=2)
```

Arguments

dims	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. dims must be specified.
lambda	The factors of CP decomposition. It is an vector. Factors lambda will be given randomly if it is NULL.
d0	d0 is the mode. The output tensor is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if the output tensor is an array with dimension dims.
dr	The user-specified rank. Default is 10.
seed_id	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank tensor characterizing the form of CP decomposition with dimension dims and factors lambda.

Value

Dn	the output mode-d0-unfolding, $D_{(d_0)}$. Or an array with dimesion dims if d0 is NULL.

See Also

gtcpsem, gtt

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
lambda <- seq(6,1,by=-1)
dr <- 5

T1 <- gtcp(dims=dims,lambda=lambda,d0=1,dr=dr)
T2 <- ttu(T1,1,2,dims)
```

gtcpsem — *Generate a low-rank semi-symmetric tensor characterizing the form of CP decomposition*

Description

This function generates a low-rank semi-symmetric tensor characterizing the form of CANDECOMP/PARAFAC (CP) decomposition with dimension `dims` and factors `lambda`. The semi-symmetric tensor means that both the mode-`r1` and mode-`r2` unfoldings are equal.

Usage

```
gtcpsem(dims, lambda=NULL, r1=1, r2=2, d0=NULL, dr=NULL, seed_id=2)
```

Arguments

`dims`	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. `dims` must be specified.
`lambda`	The factors of CP decomposition. It is an vector. Factors `lambda` will be given randomly if it is `NULL`.
`r1`	Both `r1` and `r2` are the user-specified modes, which means that both the mode-`r1` and mode-`r2` unfoldings are equal. Default is `r1 = 1` and `r2 = 2`.
`r2`	Both `r1` and `r2` are the user-specified modes, which means that both the mode-`r1` and mode-`r2` unfoldings are equal. Default is `r1 = 1` and `r2 = 2`.
`d0`	`d0` is the mode. The output tensor is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if the output tensor is an array with dimension `dims`.
`dr`	The user-specified rank. Default is `10`.
`seed_id`	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank semi-symmetric tensor characterizing the form of CP decomposition with dimension `dims` and factors `lambda`.

Value

`Dn`	the output mode-d0-unfolding, $D_{(d_0)}$. Or an array with dimesion `dims` if `d0` is `NULL`.

See Also

gtcp, gttsem

Examples

```
dims <- c(8,6,10,6,7)
N <- length(dims)
lambda <- seq(6,1,by=-1)
r1 <- 2
r2 <- 4
dr <- 5
```

```
T1 <- gtcpsem(dims=dims,lambda=lambda,r1=r1,r2=r2,d0=r1,dr=dr)
T2 <- ttu(T1,r1,r2,dims)
```

gtt *Generate a low-rank tensor characterizing the form of Tucker decomposition*

Description

This function generates a low-rank tensor characterizing the form of Tucker decomposition with dimension `dims` and core tensor `S`.

Usage

```
gtt(dims, S=NULL, d0=NULL, ranks=NULL, seed_id=2)
```

Arguments

`dims`	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. `dims` must be specified.
`S`	The core tensor. An array with dimension `dims`, or a mode-d1-unfolding of core tensor with size $n_1 \times \cdots \times n_d$. Core tensor `S` will be given randomly if it is `NULL`.
`d0`	`d0` is the mode. The output tensor is the mode-`d0` unfolding of the tensor. `d0` can be NULL (the default) if the output tensor is an array with dimension `dims`.
`ranks`	The user-specified ranks. It is a vector with length d. If `ranks` is `NULL` (the default), this function outputs a tensor without low-rank.
`seed_id`	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank tensor characterizing the form of Tucker decomposition with dimension `dims` and core tensor `S`.

Value

`Dn`	the output mode-d0-unfolding, $D_{(d_0)}$. Or an array with dimesion `dims` if `d0` is `NULL`.

See Also

gtcp, gttsem

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)

T1 = gtt(dims=dims,d0=1,ranks=ranks)
T2 <- ttu(T1,1,2,dims)
```

`gttsem` *Generate a low-rank semi-symmetric tensor characterizing the form of Tucker decomposition*

Description

This function generates a low-rank semi-symmetric tensor characterizing the form of Tucker decomposition with dimension dims and core tensor S. The semi-symmetric tensor means that both the mode-r1 and mode-r2 unfoldings are equal, where the absolute difference of r1 and r2 is restricted to no more than 3.

Usage

```
gttsem(dims, S=NULL, r1=1, r2=2, d0=NULL, ranks=NULL, seed_id=2)
```

Arguments

dims	The size of the tensor, which is a vector $(n_1, \cdots, n_d)$. dims must be specified.
S	The core tensor. An array with dimension dims, or a mode-d1-unfolding of core tensor with size $n_1 \times \cdots \times n_d$. Core tensor S will be given randomly if it is NULL.
r1	Both r1 and r2 are the user-specified modes, which means that both the mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
r2	Both r1 and r2 are the user-specified modes, which means that both the mode-r1 and mode-r2 unfoldings are equal. Default is r1 = 1 and r2 = 2.
d0	d0 is the mode. The output tensor is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if the output tensor is an array with dimension dims.
ranks	The user-specified ranks. It is a vector with length d. If ranks is NULL (the default), this function outputs a tensor without low-rank.
seed_id	A positive integer, the seed for generating the random numbers. Default is 2.

Details

This function generates a low-rank semi-symmetric tensor characterizing the form of Tucker decomposition with dimension dims and core tensor S.

Value

Dn	the output mode-d0-unfolding, $D_{(d_0)}$. Or an array with dimesion dims if d0 is NULL.

See Also

gtt, gtcpsem

Examples

```
dims <- c(8,6,8,6,7)
N <- length(dims)
ranks <- rep(2,N)
r1 <- 2
r2 <- 4

T1 <- gttsem(dims=dims,r1=r1,r2=r2,d0=r1,ranks=ranks)
T2 <- ttu(T1,r1,r2,dims)
```

---

hosvd *High-order SVD approximation of a tensor* Y *by Tucker or CP decomposition*

Description

High-order SVD approximation of a tensor Y by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition with preset rank. Alternating Least Squares algorithm is applied to Tuchker decomposition, and Tensor Power Method is applied to CP decomposition.

Usage

```
hosvd(Y=NULL, d0=NULL, dims=NULL, isCP=TRUE, ranks=NULL, dr=20,
                                      D0=NULL,isOrth=FALSE, eps=1e-6, max_step=100)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1\times\cdots\times n_d}$, where $N = n_1\times\cdots\times n_d$.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
isCP	A logical value indicating whether CP decomposition will be used. Default is TRUE.
ranks	The user-specified ranks. It is a vector with length d. Default is $(2, \cdots, 2)$.
dr	The user-specified rank for CP decomposition. It is useless if Tucker decomposition is used. Default is 20.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list($U_1 = U_1, \cdots, U_d = U_d, S = S$). By default, initial matrices are provided by random.
isOrth	A logical value indicating whether it outputs orthonognal PCs if CP decomposition is used. Default is FALSE.
eps	Convergence threshhold. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.
max_step	Maximum number of iterations. Default is 100.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-d0 unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew. It is a vector with the same length as dims if Tucker decomposition is used, or an integer if CP decomposition is used.

See Also

hosvd_dr

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)

fit <- hosvd(Y,N,dims,isCP=TRUE)
Tnew <- fit$Tnew
ranks1 <- fit$ranks
lambda <- fit$Tn[[N+1]]
U1 <- fit$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)
```

---

hosvd_dr *High-order SVD approximation of a tensor Y by Tucker or CP decomposition and selection of ranks*

Description

High-order SVD approximation of a tensor Y by Tucker decomposition or CANDECOMP/PARAFAC (CP) decomposition and selection of ranks. Alternating Least Squares algorithm is applied to Tuchker decomposition, and Tensor Power Method is applied to CP decomposition.

Usage

```
hosvd_dr(Y=NULL, d0=NULL, dims=NULL, isCP=TRUE, ranks=NULL, dr=100,
                        D0=NULL, isOrth=FALSE, eps=1e-6, max_step=100, thresh=1e-6)
```

Arguments

Y	An array with dimension dims, or a $n_{d0} \times N/n_{d0}$ numeric matrix of responses that is the mode d0-unfolding of tensor in $\mathcal{R}^{n_1\times\cdots\times n_d}$, where $N = n_1\times\cdots\times n_d$.
d0	d0 is the mode. Y is the mode-d0 unfolding of the tensor. d0 can be NULL (the default) if Y is an array with dimension dims.
dims	The size of tensor Y, which is a d-vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if Y is an array with dimension dims. If the length of dims is 2, it is the ordinary SVD decomposition of a matrix.
isCP	A logical value indicating whether CP decomposition will be used. Default is TRUE.
ranks	The user-specified ranks. It is a vector with length d. Default is $(2, \cdots, 2)$.
dr	The user-specified rank for CP decomposition. It is useless if Tucker decomposition is used. Default is 100.
D0	A user-specified list of initial matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, D0=list($U_1 = U_1, \cdots, U_d = U_d, S = S$). By default, initial matrices are provided by random.
isOrth	A logical value indicating whether it outputs orthonognal PCs if CP decomposition is used. Default is FALSE.
eps	Convergence threshhold in the inner loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.
max_step	Maximum number of iterations. Default is 100.
thresh	Convergence threshhold in the outer loop. The algorithm iterates until the relative change in any coefficient is less than eps. Default is 1e-6.

Details

This function gives a $n_{d0} \times N/n_{d0}$ matrix, which is the mode-$d0$ unfolding, and approximates Y.

Value

Tnew	Approximation of Y.
Tn	A list of estimated matrices of $U_1, U_2, \cdots, U_d$ and core tensor S, Tn=list($U_1 = U_1, \cdots, U_d = U_d, S = S$).
ranks	The ranks of estimated tensor Tnew. It is a vector with the same length as dims if Tucker decomposition is used, or an integer if CP decomposition is used.

See Also

hosvd

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)

fit_dr <- hosvd_dr(Y,N,dims,isCP=TRUE)
Tnew <- fit_dr$Tnew
ranks1 <- fit_dr$ranks
lambda <- fit_dr$Tn[[N+1]]
U1 <- fit_dr$Tn[[1]]
TNew1 <- ttu(Tnew,N,1,dims)
```

`tmp` *Modal Product*

Description

Modal product calculate the product an order d tensor S and a matrix M, that is $T = S \times_{d0} M$, $T_{(d0)} = M \cdot S_{(d0)}$, where $S_{(d0)}$ is the mode-$d0$ unfloding of the tensor S.

Usage

```
tmp(S=NULL, M=NULL, d0=NULL, dims=NULL)
```

Arguments

`S`	An order d tensor with dimension `dims`, where `dims` is $(n_1, \cdots, n_d)$.
`M`	A matrix with n_{d0} columns and K rows. The modal product produce a new order d tensor with dimension `dims` replaced the `d0` dimnesion by K.
`dims`	The dimension of tensor `Y`, which is a d-vector $(n_1, \cdots, n_d)$.
`d0`	`d0` is the mode. M multiplies the mode-`d0` unfoldings of `S`.

Details

This function gives the modal product of tensor S and matrix M, which is the product of M and the mode-`d0` unfolding of tensor S, that is $S \times_{d0} M = MS_{(d0)}$, where $S_{(d0)}$ is the mode-$d0$ unfloding of the tensor S. The modal product produce a new order d tensor with dimension `dims` replaced the `d0` dimnesion by K.

Value

Tnew Product of S and M.

See Also

TransUnfoldingsT

Examples

```
dims <- c(8,8,10,10,6)
N <- length(dims)
ranks <- rep(2,N)
S0 <- matrix(runif(prod(ranks),3,7),ranks[N])
T1 <- matrix(rnorm(dims[1]*ranks[1]),nrow = dims[1])
tmp <- qr.Q(qr(T1))
for(k in 2:(N-1)){
  T1 <- matrix(rnorm(dims[k]*ranks[k]),nrow = dims[k])
  tmp <- kronecker(qr.Q(qr(T1)),tmp)
}
T1 <- matrix(rnorm(dims[N]*ranks[N]),nrow = dims[N])
U <- qr.Q(qr(T1))
Y <- U%*%S0%*%t(tmp)
Y1 <- array(ttu(Y,N,1,dims),dims)
M <- matrix(1:(4*dims[3]),4)

X1 <- tmp(Y1,M,3,dims)
X <- ttu(matrix(X1,dims[1]),1,N,dims)

dim(Y1)
dim(X1)
print(Y[,1])
print(X[,1])
```

---

ttu *Transfer a tensor's modal unfoldings to another.*

Description

Transfer a tensor's modal unfoldings to another.

Usage

```
ttu(S=NULL, d1=NULL, d2=0, dims=NULL)
```

Arguments

S An array with dimension dims, or a mode-d1-unfolding of a tensor with size $n_1 \times \cdots \times n_d$.

d1 An integer, the mode of unfolding $S_{(d_1)}$. d1 can be NULL (the default) if S is an array with dimension dims.

d2	An integer, the mode of output unfolding $S_{(d_2)}$. It transfers S to an array with dimension dims if d2=0. The default is 0.
dims	The size of tensor S, which is a vector $(n_1, \cdots, n_d)$. dims can be NULL (the default) if S is an array with dimension dims.

Details

This function transfers an input mode-d1-unfolding $S_{(d_1)}$ to mode-d2-unfolding $S_{(d_2)}$

Value

Td2	the output mode-d2-unfolding, $S_{(d_2)}$.

Examples

```
T1 <- matrix(1:24,nrow = 4) # A tensor unfolding with size 4*6
T2 <- ttu(T1,1,2,c(4,3,2))

T0 <- ttu(T2,2,dims=c(4,3,2))
```

Index