Package ‘nnTensor’

June 24, 2021

Type Package

Title Non-Negative Tensor Decomposition

Version 1.1.0

Date 2021-06-22

Suggests testthat

Depends R (>= 3.4.0)

Imports methods, fields, rTensor, plot3D, tagcloud, ggplot2


License Artistic-2.0

URL https://github.com/rikenbit/nnTensor

NeedsCompilation no

Author Koki Tsuyuzaki [aut, cre],
        Manabu Ishii [aut],
        Itoshi Nikaido [aut]

Maintainer Koki Tsuyuzaki <k.t.the-answer@hotmail.co.jp>

Repository CRAN

Date/Publication 2021-06-24 09:20:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>nnTensor-package</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>GabrielNMF</td>
<td>4</td>
</tr>
<tr>
<td>jNMF</td>
<td>5</td>
</tr>
<tr>
<td>NMF</td>
<td>7</td>
</tr>
<tr>
<td>NTD</td>
<td>10</td>
</tr>
<tr>
<td>NTF</td>
<td>12</td>
</tr>
<tr>
<td>plot.NMF</td>
<td>13</td>
</tr>
</tbody>
</table>
Description


Details

The DESCRIPTION file:

<table>
<thead>
<tr>
<th>Package:</th>
<th>nnTensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Title:</td>
<td>Non-Negative Tensor Decomposition</td>
</tr>
<tr>
<td>Version:</td>
<td>1.1.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2021-06-22</td>
</tr>
<tr>
<td>Authors@R:</td>
<td>c(person(&quot;Koki&quot;, &quot;Tsuyuzaki&quot;, role = c(&quot;aut&quot;, &quot;cre&quot;), email = &quot;<a href="mailto:k.t.the-answer@hotmail.co.jp">k.t.the-answer@hotmail.co.jp</a>&quot;), person(&quot;Manabu Ishii&quot;, role = &quot;aut&quot;), person(&quot;Itoshi Nikaido&quot;, role = &quot;aut&quot;))</td>
</tr>
<tr>
<td>Suggests:</td>
<td>testthat</td>
</tr>
<tr>
<td>Depends:</td>
<td>R (&gt;= 3.4.0)</td>
</tr>
<tr>
<td>Imports:</td>
<td>methods, fields, rTensor, plot3D, tagcloud, ggplot2</td>
</tr>
<tr>
<td>License:</td>
<td>Artistic-2.0</td>
</tr>
<tr>
<td>URL:</td>
<td><a href="https://github.com/rikenbit/nnTensor">https://github.com/rikenbit/nnTensor</a></td>
</tr>
<tr>
<td>Author:</td>
<td>Koki Tsuyuzaki [aut, cre], Manabu Ishii [aut], Itoshi Nikaido [aut]</td>
</tr>
<tr>
<td>Maintainer:</td>
<td>Koki Tsuyuzaki <a href="mailto:k.t.the-answer@hotmail.co.jp">k.t.the-answer@hotmail.co.jp</a></td>
</tr>
</tbody>
</table>

Index of help topics:

| GabrielNMF       | Gabriel-type Bi-Cross-Validation for Non-negative Matrix Factorization |
| NMF              | Non-negative Matrix Factorization Algorithms (NMF) |
| NTD              | Non-negative Tucker Decomposition Algorithms (NTD) |
| NTF              | Non-negative CP Decomposition Algorithms (NTF) |
| jNMF             | Joint Non-negative Matrix Factorization Algorithms (jNMF) |
nnTensor-package

nnTensor-package Non-Negative Tensor Decomposition
plot.NMF Plot function of the result of NMF function
plotTensor3D Plot function for visualization of tensor data structure
recTensor Tensor Reconstruction from core tensor (S) and factor matrices (A)
siNMF Simultaneous Non-negative Matrix Factorization Algorithms (siNMF)
toyModel Toy model data for using NMF, NTF, and NTD

Author(s)
NA
Maintainer: NA

References


Anh Hyu Phan et. al. (2011). Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*


Xiaoxu Han. (2007). CANCER MOLECULAR PATTERN DISCOVERY BY SUBSPACE CONSENSUS KERNEL CLASSIFICATION


Chunxuan Shao et al. (2017). Robust classification of single-cell transcriptome data by nonnegative matrix factorization. *Bioinformatics*

Paul Fogel (2013). Permutted NMF: A Simple Algorithm Intended to Minimize the Volume of the Score Matrix

Philip M. Kim et al. (2003). Subsystem Identification Through Dimensionality Reduction of Large-Scale Gene Expression Data. *Genome Research*


See Also
toyModel,NMF,NTF,NTD,recTensor,plotTensor3D

Examples

```r
ls("package:nnTensor")
```

---

### Description

The input data is assumed to be non-negative matrix. GabrielNMF divides the input file into four matrices (A, B, C, and D) and perform cross validation by the prediction of A from the matrices B, C, and D.

### Usage

```r
GabrielNMF(X, J = 3, nx = 5, ny = 5, ...)
```

### Arguments

- `X` The input matrix which has N-rows and M-columns.
- `J` The number of low-dimension (J < N, M).
- `nx` The number of hold-out in row-wise direction (2 < nx < N).
- `ny` The number of hold-out in row-wise direction (2 < ny < M).
- `...` Other parameters for NMF function.
Value

TestRecError: The reconstruction error calculated by Gabriel-style Bi-Cross Validation.

Author(s)

Koki Tsuyuzaki

References


Examples

```r
if(interactive()){
  # Test data
  matdata <- toyModel(model = "NMF")

  # Bi-Cross-Validation
  BCV <- rep(0, length=5)
  names(BCV) <- 2:6
  for(j in seq(BCV)){
    print(j+1)
    BCV[j] <- mean(GabrielNMF(matdata, J=j+1, nx=2, ny=2)$TestRecError)
  }
  proper.rank <- as.numeric(names(BCV)[which(BCV == min(BCV))])

  # NMF
  out <- NMF(matdata, J=proper.rank)
}
```

---

**jNMF**

*Joint Non-negative Matrix Factorization Algorithms (jNMF)*

Description

The input data objects are assumed to be non-negative matrices. jNMF decompose the matrices to two low-dimensional factor matrices simultaneously.

Usage

```r
jNMF(X, M=NULL, initW=NULL, initV=NULL, initH=NULL, fixW=FALSE, fixV=FALSE,
fixH=FALSE,
L1_W=1e-10, L1_V=1e-10, L1_H=1e-10,
L2_W=1e-10, L2_V=1e-10, L2_H=1e-10,
J = 3, w=NULL, algorithm = c("Frobenius", "KL", "IS", "PLTF"),
p=1, thr = 1e-10, num.iter = 100, viz = FALSE, figdir = NULL, verbose = FALSE)
```
Arguments

X  A list containing input matrices (X_k, <N*Mk>, k=1..K).

M  A list containing the mask matrices (X_k, <N*Mk>, k=1..K). If the input matrix has missing values, specify the element as 0 (otherwise 1).

initW  The initial values of factor matrix W, which has N-rows and J-columns (Default: NULL).

initV  A list containing the initial values of multiple factor matrices (V_k, <N*J>, k=1..K, Default: NULL).

initH  A list containing the initial values of multiple factor matrices (H_k, <Mk*J>, k=1..K, Default: NULL).

fixW  Whether the factor matrix W is updated in each iteration step (Default: FALSE).

fixV  Whether the factor matrices Vk are updated in each iteration step (Default: FALSE).

fixH  Whether the factor matrices Hk are updated in each iteration step (Default: FALSE).

L1_W  Parameter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.

L1_V  Parameter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.

L1_H  Parameter for L1 regularitation (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.

L2_W  Parameter for L2 regularitation (Default: 1e-10).

L2_V  Parameter for L2 regularitation (Default: 1e-10).

L2_H  Parameter for L2 regularitation (Default: 1e-10).

J  Number of low-dimension (J < N, Mk).

w  Weight vector (Default: NULL)

algorithm  Divergence between X and X_bar. "Frobenius", "KL", and "IS" are available (Default: "KL").

p  The parameter of Probabilistic Latent Tensor Factorization (p=0: Frobenius, p=1: KL, p=2: IS)

thr  When error change rate is lower than thr, the iteration is terminated (Default: 1E-10).

num.iter  The number of iteration step (Default: 100).

viz  If viz == TRUE, internal reconstructed matrix can be visualized.

figdir  the directory for saving the figure, when viz == TRUE.

verbose  If verbose == TRUE, Error change rate is generated in console window.
Value

W : A matrix which has N-rows and J-columns (J < N, Mk). V : A list which has multiple elements containing N-rows and J-columns (J < N, Mk). H : A list which has multiple elements containing Mk-rows and J-columns matrix (J < N, Mk). RecError : The reconstruction error between data matrix and reconstructed matrix from W and H. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References


Examples

```r
matdata <- toyModel(model = "siNMF_Hard")
out <- jNMF(matdata, J=2, num.iter=2)
```

Description

The input data is assumed to be non-negative matrix. NMF decompose the matrix to two low-dimensional factor matrices. This function is also used as initialization step of tensor decomposition (see also NTF and NTD).
**Usage**

NMF(X, M=NULL, initU=NULL, initV=NULL, fixU=FALSE, fixV=FALSE,
L1_U=1e-10, L1_V=1e-10, L2_U=1e-10, L2_V=1e-10, J = 3,
rank.method=c("all", "ccc", "dispersion", "rss", "evar", "residuals",
"sparseness.basis", "sparseness.coeff", "sparseness2.basis",
"sparseness2.coeff", "norm.info.gain.basis", "norm.info.gain.coeff",
"singular", "volume", "condition"), runtime=30,
"Alpha", "Beta", "PGD", "HALS", "GCD"), Alpha = 1, Beta = 2,
eta = 1e-04, thr1 = 1e-10, thr2 = 1e-10, tol = 1e-04,
num.iter = 100, viz = FALSE, figdir = NULL, verbose = FALSE)

**Arguments**

- **X**: The input matrix which has N-rows and M-columns.
- **M**: The mask matrix which has N-rows and M-columns. If the input matrix has missing values, specify the element as 0 (otherwise 1).
- **initU**: The initial values of factor matrix U, which has N-rows and J-columns (Default: NULL).
- **initV**: The initial values of factor matrix V, which has M-rows and J-columns (Default: NULL).
- **fixU**: Whether the factor matrix U is updated in each iteration step (Default: FALSE).
- **fixV**: Whether the factor matrix V is updated in each iteration step (Default: FALSE).
- **L1_U**: Parameter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
- **L1_V**: Parameter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
- **L2_U**: Parameter for L2 regularization (Default: 1e-10).
- **L2_V**: Parameter for L2 regularization (Default: 1e-10).
- **J**: The number of low-dimension (J < N, M). If a numerical vector is specified (e.g. 2:6), the appropriate rank is estimated.
- **rank.method**: The rank estimation method (Default: "all"). Only if the J option is specified as a numerical vector longer than two, this option will be active.
- **runtime**: The number of trials to estimate rank (Default: 10).
- **Alpha**: The parameter of Alpha-divergence.
- **Beta**: The parameter of Beta-divergence.
- **eta**: The stepsize for PGD algorithm (Default: 0.0001).
- **thr1**: When error change rate is lower than thr1, the iteration is terminated (Default: 1E-10).
- **thr2**: If the minus-value is generated, replaced as thr2 (Default: 1E-10). This value is used within the internal function .positive().
tol          The tolerance parameter used in GCD algorithm.
num.iter     The number of iteration step (Default: 100).
viz          If viz == TRUE, internal reconstructed matrix can be visualized.
figdir       The directory for saving the figure, when viz == TRUE.
verbose      If verbose == TRUE, Error change rate is generated in console window.

Value

U : A matrix which has N-rows and J-columns (J < N, M). V : A matrix which has M-rows and J-columns (J < N, M). J : The number of dimension (J < N, M). RecError : The reconstruction error between data tensor and reconstructed tensor from U and V. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error. Trial : All the results of the trials to estimate the rank. Runtime : The number of the trials to estimate the rank. RankMethod : The rank estimation method.

Author(s)

Koki Tsuyuzaki

References


Examples

if(interactive()){
  # Test data
  matdata <- toyModel(model = "NMF")

  # Simple usage
  out <- NMF(matdata, J=5)

  # Rank estimation mode (single method)
  out2 <- NMF(matdata, J=2:10, rank.method="ccc", runtime=3)
  plot(out2)

  # Rank estimation mode (all method)
  out3 <- NMF(matdata, J=2:10, rank.method="all", runtime=10)
  plot(out3)
}
Non-negative Tucker Decomposition Algorithms (NTD)

Description

The input data is assumed to be non-negative tensor. NTD decomposes the tensor to the dense core tensor (S) and low-dimensional factor matrices (A).

Usage

```r
NTD(X, M=NULL, initS=NULL, initA=NULL, fixS=FALSE, fixA=FALSE,
    L1_A=1e-10, L2_A=1e-10,
    rank = c(3, 3, 3), modes = 1:3,
    algorithm = c("Frobenius", "KL", "IS", "Pearson", "Hellinger",
                  "Neyman", "HALS", "Alpha", "Beta"), init = c("NMF", "ALS", "Random"),
    Alpha = 1,
    Beta = 2, thr = 1e-10, num.iter = 100, viz = FALSE,
    figdir = NULL, verbose = FALSE)
```

Arguments

- **X**: The input tensor which has I1, I2, and I3 dimensions.
- **M**: The mask tensor which has I1, I2, and I3 dimensions. If the mask tensor has missing values, specify the element as 0 (otherwise 1).
- **initS**: The initial values of core tensor which has J1, J2, and J3 dimensions (Default: NULL).
- **initA**: A list containing the initial values of multiple factor matrices (A_k, <Ik*Jk>, k=1..K, Default: NULL).
- **fixS**: Whether the core tensor S is updated in each iteration step (Default: FALSE).
- **fixA**: Whether the factor matrices Ak are updated in each iteration step (Default: FALSE).
- **L1_A**: Parameter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
- **L2_A**: Parameter for L2 regularization (Default: 1e-10).
- **rank**: The number of low-dimension in each mode (J1, J2, J3, J1<I1, J2<I2, J3 < I3) (Default: c(3,3,3)).
- **modes**: The vector of the modes on which to perform the decomposition (Default: 1:3 <all modes>.
- **init**: The initialization algorithms. "NMF", "ALS", and "Random" are available (Default: "NMF").
- **Alpha**: The parameter of Alpha-divergence.
Beta: The parameter of Beta-divergence.

thr: When error change rate is lower than thr1, the iteration is terminated (Default: 1E-10).

num.iter: The number of iteration step (Default: 100).

viz: If viz == TRUE, internal reconstructed tensor can be visualized.

figdir: the directory for saving the figure, when viz == TRUE (Default: NULL).

verbose: If verbose == TRUE, Error change rate is generated in console window.

Value

S: Tensor object, which is defined as S4 class of rTensor package. A: A list containing three factor matrices. RecError: The reconstruction error between data tensor and reconstructed tensor from S and A. TrainRecError: The reconstruction error calculated by training set (observed values specified by M). TestRecError: The reconstruction error calculated by test set (missing values specified by M). RelChange: The relative change of the error.

Author(s)

Koki Tsuyuzaki

References


Anh Hyu Phan et. al. (2011). Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*

See Also

plotTensor3D

Examples

tensordata <- toyModel(model = "Tucker")
out <- NTD(tensordata, rank=c(2,2,2), algorithm="Frobenius",
           init="Random", num.iter=2)
Non-negative CP Decomposition Algorithms (NTF)

Description

The input data is assumed to be non-negative tensor. NTF decompose the tensor to the diagonal core tensor $(S)$ and low-dimensional factor matrices $(A)$.

Usage

```r
NTF(X, M=NULL, initA=NULL, fixA=FALSE, L1_A=1e-10, L2_A=1e-10, rank = 3,
      "HALS", "Alpha-HALS", "Beta-HALS", "Alpha", "Beta"),
    init = c("NMF", "ALS", "Random"), Alpha = 1,
    Beta = 2, thr = 1e-10, num.iter = 100, viz = FALSE,
    figdir = NULL, verbose = FALSE)
```

Arguments

- **X**: The input tensor which has $I_1, I_2, \text{and } I_3$ dimensions.
- **M**: The mask tensor which has $I_1, I_2, \text{and } I_3$ dimensions. If the mask tensor has missing values, specify the element as 0 (otherwise 1).
- **initA**: A list containing the initial values of multiple factor matrices ($A_k$, $<I_k*J_k>$, $k=1..K$, Default: NULL).
- **fixA**: Whether the factor matrices $A_k$ are updated in each iteration step (Default: FALSE).
- **L1_A**: Parameter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
- **L2_A**: Parameter for L2 regularization (Default: 1e-10).
- **rank**: The number of low-dimension in each mode ($J_1=J_2=J_3, J_1<I_1, J_2<I_2, J_3 < I_3$) (Default: 3).
- **init**: The initialization algorithms. "NMF", "ALS", and "Random" are available (Default: "NMF").
- **Alpha**: The parameter of Alpha-divergence.
- **Beta**: The parameter of Beta-divergence.
- **thr**: When error change rate is lower than thr1, the iteration is terminated (Default: 1E-10).
- **num.iter**: The number of interation step (Default: 100).
- **viz**: If viz == TRUE, internal reconstructed tensor can be visualized.
- **figdir**: the directory for saving the figure, when viz == TRUE (Default: NULL).
- **verbose**: If verbose == TRUE, Error change rate is generated in console window.
Value

S : Tensor object, which is defined as S4 class of rTensor package. A : A list containing three factor matrices. RecError : The reconstruction error between data tensor and reconstructed tensor from S and A. TrainRecError : The reconstruction error calculated by training set (observed values specified by M). TestRecError : The reconstruction error calculated by test set (missing values specified by M). RelChange : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References


See Also

plotTensor3D

Examples

tensordata <- toyModel(model = "CP")
out <- NTF(tensordata, rank=3, algorithm="Beta-HALS", num.iter=2)

plot.NMF

Plot function of the result of NMF function

Description

Only if J is specified as a vector longer than 1, this function will be active.

Usage

plot(x, ...)

Arguments

x The result of NMF function (NMF class).
... Optional parameter for plot.
plotTensor3D

Value
A ggplot will be generated.

Author(s)
Koki Tsuyuzaki

References

Xiaoxu Han. (2007). CANCER MOLECULAR PATTERN DISCOVERY BY SUBSPACE CONSENSUS KERNEL CLASSIFICATION


Examples

```r
plot.NMF
```

```r
plotTensor3D
```

Plot function for visualization of tensor data structure

Description
Combined with recTensor function and the result of NTF or NTD, the reconstructed tensor structure can be visualized.

Usage

```r
plotTensor3D(X = NULL)
```
Arguments

\[ X \] Tensor object, which is defined as S4 class of rTensor package.

Author(s)

Koki Tsuyuzaki

Examples

tensordata <- toyModel(model = "CP")
out <- NTF(tensordata, rank=3, algorithm="Beta-HALS", num.iter=2)
tmp <- tempdir()
png(filename=paste0(tmp, "/NTF.png"))
plotTensor3D(recTensor(out$S, out$A))
dev.off()

---

**recTensor**  
Tensor Reconstruction from core tensor (S) and factor matrices (A)

Description

Combined with plotTensor3D function and the result of NTF or NTD, the reconstructed tensor structure can be visualized.

Usage

recTensor(S = NULL, A = NULL, idx = 1:3, reverse = FALSE)

Arguments

\[ S \] Tensor object, which is defined as S4 class of rTensor package.

\[ A \] A list containing three factor matrices.

\[ idx \] The direction of mode-n multiplication (Default: 1:3). For example idx=1 is defined. S x_1 A is calculated (x_1 : mode-1 multiplication).

\[ reverse \] If reverse = TRUE, t(A[[n]]) is multiplicated to S (Default: FALSE).

Value

Tensor object, which is defined as S4 class of rTensor package.

Author(s)

Koki Tsuyuzaki
siNMF

Simultaneous Non-negative Matrix Factorization Algorithms (siNMF)

Description

The input data objects are assumed to be non-negative matrices. siNMF decompose the matrices to two low-dimensional factor matrices simultaneously.

Usage

```r
siNMF(X, M=NULL, initW=NULL, initH=NULL, fixW=FALSE, fixH=FALSE,
L1_W=1e-10, L1_H=1e-10, L2_W=1e-10, L2_H=1e-10, J = 3,
w=NULL, algorithm = c("Frobenius", "KL", "IS", "PLTF"), p=1,
thr = 1e-10, num.iter = 100,
viz = FALSE, figdir = NULL, verbose = FALSE)
```

Arguments

- **X**: A list containing the input matrices (X_k, <N*M_k>, k=1..K).
- **M**: A list containing the mask matrices (X_k, <N*M_k>, k=1..K). If the input matrix has missing values, specify the element as 0 (otherwise 1).
- **initW**: The initial values of factor matrix W, which has N-rows and J-columns (Default: NULL).
- **initH**: A list containing the initial values of multiple factor matrices (H_k, <M_k*J>, k=1..K, Default: NULL).
- **fixW**: Whether the factor matrix W is updated in each iteration step (Default: FALSE).
- **fixH**: Whether the factor matrices H_k are updated in each iteration step (Default: FALSE).
- **L1_W**: Parameter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
- **L1_H**: Parameter for L1 regularization (Default: 1e-10). This also works as small positive constant to prevent division by zero, so should be set as 0.
- **L2_W**: Parameter for L2 regularization (Default: 1e-10).
- **L2_H**: Parameter for L2 regularization (Default: 1e-10).
- **J**: Number of low-dimension (J < N, M_k).

See Also

`Tensor-class, NTF, NTD`

Examples

```r
tensordata <- toyModel(model = "CP")
out <- NTF(tensordata, rank=3, algorithm="Beta-HALS", num.iter=2)
rec <- recTensor(out$S, out$A)
```
The parameter of Probabilistic Latent Tensor Factorization (\(p=0\): Frobenius, \(p=1\): KL, \(p=2\): IS)  

When error change rate is lower than \(\text{thr}\), the iteration is terminated (Default: \(1E^{-10}\)).  

The number of iteration step (Default: 100).  

If \(\text{viz} == \text{TRUE}\), internal reconstructed matrix can be visualized.  

the directory for saving the figure, when \(\text{viz} == \text{TRUE}\).  

If \(\text{verbose} == \text{TRUE}\), Error change rate is generated in console windo.

Value

\(W\) : A matrix which has \(N\)-rows and \(J\)-columns \((J < N, M_k)\).  
\(H\) : A list which has multiple elements containing \(M_k\)-rows and \(J\)-columns matrix \((J < N, M_k)\).  
\(\text{RecError}\) : The reconstruction error between data matrix and reconstructed matrix from \(W\) and \(H\).  
\(\text{TrainRecError}\) : The reconstruction error calculated by training set (observed values specified by \(M\)).  
\(\text{TestRecError}\) : The reconstruction error calculated by test set (missing values specified by \(M\)).  
\(\text{RelChange}\) : The relative change of the error.

Author(s)

Koki Tsuyuzaki

References


Examples

```r
matdata <- toyModel(model = "siNMF_Easy")
out <- siNMF(matdata, J=2, num.iter=2)
```
toymodel

Toy model data for using NMF, NTF, and NTD

Description

The data is used for confirming the algorithm are properly working.

Usage

```r
toyModel(model = "CP", seeds=123)
```

Arguments

- `model` Single character string is specified. "NMF", "CP", and "Tucker" are available (Default: "CP").
- `seeds` Random number for setting set.seeds in the function (Default: 123).

Value

If model is specified as "NMF", a matrix is generated. Otherwise, a tensor is generated.

Author(s)

Koki Tsuyuzaki

See Also

NMF, NTF, NTD

Examples

```r
matdata <- toyModel(model = "NMF", seeds=123)
tensordata1 <- toyModel(model = "CP", seeds=123)
tensordata2 <- toyModel(model = "Tucker", seeds=123)
```
Index

* methods
  GabrielNMF, 4
  jNMF, 5
  NMF, 7
  NTD, 10
  NTF, 12
  plot.NMF, 13
  plotTensor3D, 14
  recTensor, 15
  siNMF, 16
  toyModel, 18

* package
  nnTensor-package, 2

GabrielNMF, 4

jNMF, 5

NMF, 4, 7, 18
nnTensor (nnTensor-package), 2
nnTensor-package, 2
NTD, 4, 10, 16, 18
NTF, 4, 12, 16, 18

plot (plot.NMF), 13
plot.NMF, 13
plotTensor3D, 4, 11, 13, 14

recTensor, 4, 15

siNMF, 16

toyModel, 4, 18