

Package: mwTensor (via r-universe)

October 26, 2026

Type Package

Title Multi-Way Component Analysis

Version 1.2.2

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports methods, MASS, rTensor, nnTensor, ccTensor, iTensor, igraph

Description For single tensor data, any matrix factorization method can be specified the matricised tensor in each dimension by Multi-way Component Analysis (MWCA). An originally extended MWCA is also implemented to specify and decompose multiple matrices and tensors simultaneously (CoupledMWCA). See the reference section of GitHub README.md
<<https://github.com/rikenbit/mwTensor>>, for details of the methods.

License MIT + file LICENSE

URL <https://github.com/rikenbit/mwTensor>

Config/pak/sysreqs cmake libfreetype6-dev libglpk-dev libglu1-mesa-dev make texlive libicu-dev libpng-dev libuv1-dev libxml2-dev libgl1-mesa-dev zlib1g-dev

Repository <https://rikenbit.r-universe.dev>

Date/Publication 2026-05-07 05:37:38 UTC

RemoteUrl <https://github.com/rikenbit/mwtensor>

RemoteRef HEAD

RemoteSha 4ea233ec598ee64560880aa58ab09de383205729

Contents

mwTensor-package	2
checkCoupledMWCA	3
compileMWCAProgram	5

CoupledMWCA	5
CoupledMWCAInit-class	6
CoupledMWCAParams-class	7
CoupledMWCAResult-class	9
defaultCoupledMWCAParams	10
defaultMWCAParams	11
executeMWCAProgram	12
initCoupledMWCA	13
MWCA	14
MWCAParams-class	15
MWCAProgram	16
MWCAResult-class	17
myALS_SVD	18
myCX	19
myICA	20
myNMF	21
mySVD	22
plotTensor3Ds	22
RefinedFactor-class	23
refineFactor	24
toyModel	25
validateMWCAProgram	26

Index	27
--------------	-----------

mwTensor-package	<i>Multi-Way Component Analysis</i>
------------------	-------------------------------------

Description

For single tensor data, any matrix factorization method can be specified the matricised tensor in each dimension by Multi-way Component Analysis (MWCA). An originally extended MWCA is also implemented to specify and decompose multiple matrices and tensors simultaneously (CoupledMWCA). See the reference section of GitHub README.md <<https://github.com/rikenbit/mwTensor>>, for details of the methods.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

Koki Tsuyuzaki [aut, cre]

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

References

- Andrzej Cichocki et al., (2016). Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions
- Andrzej Cichocki et al., (2015). Tensor Decompositions for Signal Processing Applications, *IEEE SIGNAL PROCESSING MAGAZINE*
- Gene H. Golub et al., (2012). Matrix Computation (Johns Hopkins Studies in the Mathematical Sciences), *Johns Hopkins University Press*
- Madeleine Udell et al., (2016). Generalized Low Rank Models, *Foundations and Trends in Machine Learning*, 9(1).
- Andrzej CICHOCK, et. al., (2009). Nonnegative Matrix and Tensor Factorizations.
- A. Hyvarinen. (1999). Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks*, 10(3), 626-634.
- Petros Drineas et al., (2008). Relative-Error CUR Matrix Decompositions, *SIAM Journal on Matrix Analysis and Applications*, 30(2), 844-881.

See Also

[mySVD](#), [myALS_SVD](#), [myNMF](#), [myICA](#), [myCX](#), [MWCA](#), [CoupledMWCA](#), [checkCoupledMWCA](#), [initCoupledMWCA](#), [refineFactor](#), [MWCAProgram](#), [plotTensor3Ds](#)

Examples

```
ls("package:mwTensor")
```

checkCoupledMWCA

Static Validation of CoupledMWCA Parameters

Description

Validates a CoupledMWCAParams object without running optimization. Collects all errors and warnings instead of stopping on the first failure. This is useful for pre-validating candidate decomposition problems before committing to a potentially expensive CoupledMWCA run.

Usage

```
checkCoupledMWCA(params)
```

Arguments

params A CoupledMWCAParams object.

Value

A list with the following components:

ok Logical. TRUE if no errors were found.

errors Character vector of error messages (empty if ok is TRUE).

warnings Character vector of warning messages.

normalized_params The input params if valid, or NULL if errors exist.

summary A single human-readable summary string.

Author(s)

Koki Tsuyuzaki

References

Andrzej Cichocki et al., (2016). Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part I Low-Rank Tensor Decompositions

Andrzej Cichocki et al., (2015). Tensor Decompositions for Signal Processing Applications, *IEEE SIGNAL PROCESSING MAGAZINE*

See Also

[CoupledMWCA](#), [defaultCoupledMWCAParams](#), [CoupledMWCAParams-class](#).

Examples

```
if(interactive()){
  # Test data
  Xs <- toyModel("coupled_CP_Easy")
  common_model <- list(
    X1=list(I1="A1", I2="A2"),
    X2=list(I2="A2", I3="A3", I4="A4"),
    X3=list(I4="A4", I5="A5"))
  params <- defaultCoupledMWCAParams(Xs, common_model)
  # Validate without running optimization
  result <- checkCoupledMWCA(params)
  result$ok      # TRUE
  result$errors  # character(0)
  result$summary # "Validation passed"
}
```

compileMWCAProgram	<i>Compile an MWCAProgram to Solver Parameters</i>
--------------------	--

Description

Converts a validated, non-recursive MWCAProgram into MWCAParams (single block) or CoupledMWCAParams (multiple blocks). Programs with refinements cannot currently be compiled.

Usage

```
compileMWCAProgram(program, Xs, ...)
```

Arguments

program	An MWCAProgram object.
Xs	Named list of input arrays, with names matching the program's block names.
...	Additional parameters: thr, verbose, common_coretype, specific_coretype, etc.

Value

An MWCAParams or CoupledMWCAParams object.

Author(s)

Koki Tsuyuzaki

See Also

[MWCAProgram](#), [validateMWCAProgram](#), [MWCA](#), [CoupledMWCA](#).

CoupledMWCA	<i>Coupled Multi-way Component Analysis (CoupledMWCA)</i>
-------------	---

Description

The input is assumed to be a CoupledMWCAParams or CoupledMWCAInit object. When a [CoupledMWCAInit](#) object (produced by [initCoupledMWCA](#)) is passed, its pre-computed factor matrices are used as the starting point for optimization.

Usage

```
CoupledMWCA(params)
```

Arguments

params A CoupledMWCAParams or CoupledMWCAInit object.

Value

CoupledMWCAResult object.

Author(s)

Koki Tsuyuzaki

See Also

[CoupledMWCAParams-class](#) and [CoupledMWCAResult-class](#).

Examples

```
if(interactive()){
  # Test data (multiple arrays)
  Xs=list(
    X1=array(runif(7*4), dim=c(7,4)),
    X2=array(runif(4*5*6), dim=c(4,5,6)),
    X3=array(runif(6*8), dim=c(6,8)))
  # Setting of factor matrices
  common_model=list(
    X1=list(I1="A1", I2="A2"),
    X2=list(I2="A2", I3="A3", I4="A4"),
    X3=list(I4="A4", I5="A5"))
  # Default Parameters
  params <- defaultCoupledMWCAParams(Xs=Xs, common_model=common_model)
  # Perform Coupled MWCA
  out <- CoupledMWCA(params)
}
```

CoupledMWCAInit-class *Class "CoupledMWCAInit"*

Description

Container for CoupledMWCA initialization results produced by [initCoupledMWCA](#).

Slots

params The CoupledMWCAParams object used for initialization.

common_factors Named list of common factor matrices.

common_cores List of common core tensors.

specific_factors Named list of specific factor matrices.

specific_cores List of specific core tensors.
init_policy Character string indicating the initialization policy used.
seed The random seed used, or NULL.

See Also

[initCoupledMWCA](#), [CoupledMWCAParams-class](#).

CoupledMWCAParams-class

Class "CoupledMWCAParams"

Description

The parameter object to be specified against CoupledMWCA function.

Objects from the Class

Objects can be created by calls of the form `new("CoupledMWCAParams", ...)`.

Slots

MWCAParams has four settings as follows. For each setting, the list must have the same structure.

1. *Data-wise setting* Each item must be a list object that is as long as the number of data and is named after the data.

A list containing multiple high-dimensional arrays.

mask: A list containing multiple high-dimensional arrays, in which 0 or 1 values are filled to specify the missing elements.

pseudocount: The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).

weights: A list containing multiple high-dimensional arrays, in which some numeric values are specified to weigh each data.

2. *Common Model setting* Each item must be a nested list object that is as long as the number of data and is named after the data.

common_model: Each element of the list must be a list corresponding the dimension name of data and common factor matrices name.

3. *Common Factor matrix-wise setting* Each item must be a list object that is as long as the number of common factor matrices and is named after the factor matrices.

common_initial: The initial values of common factor matrices. If nothing is specified, random matrices are used.

common_algorithms: Algorithms used to decompose the matrixed tensor in each mode.

common_iteration: The number of iterations.

common_decomp: If FALSE is specified, unit matrix is used as the common factor matrix.

common_fix: If TRUE is specified, the common factor matrix is not updated in the iteration.

common_dims: The lower dimension of each common factor matrix.

common_transpose: Whether the common factor matrix is transposed to calculate core tensor.

common_coretype: If "CP" is specified, all the core tensors become diagonal core tensors. If "Tucker" is specified, all the core tensors become dense core tensors.

4. *Specific Model setting* Each item must be a nested list object that is as long as the number of data and is named after the data.

specific_model: Each element of the list must be a list corresponding the dimension name of data and data specific factor matrices name.

5. *Specific Factor matrix-wise setting* Each item must be a list object that is as long as the number of data specific factor matrices and is named after the factor matrices.

specific_initial: The initial values of data specific factor matrices. If nothing is specified, random matrices are used.

specific_algorithms: Algorithms used to decompose the matricised tensor in each mode.

specific_iteration: The number of iterations.

specific_decomp: If FALSE is specified, unit matrix is used as the data specific factor matrix.

specific_fix: If TRUE is specified, the data specific factor matrix is not updated in the iteration.

specific_dims: The lower dimension of each data specific factor matrix.

specific_transpose: Whether the data specific factor matrix is transposed to calculate core tensor.

specific_coretype: If "CP" is specified, all the core tensors become diagonal core tensors. If "Tucker" is specified, all the core tensors become dense core tensors.

6. *Other option* Each item must to be a vector of length 1.

specific: Whether data specific factor matrices are also calculated.

thr: The threshold to stop the iteration. The higher the value, the faster the iteration will stop.

viz: Whether the output is visualized.

figdir: When viz=TRUE, whether the plot is output in the directory.

verbose: Whether the process is monitored by verbose messages.

Methods

CoupledMWCA Function to perform CoupledMWCA.

See Also

[CoupledMWCAResult-class](#), [CoupledMWCA](#)

CoupledMWCAResult-class

Class "*CoupledMWCAResult*"

Description

The result object generated by CoupledMWCA function.

Slots

weights: weights of CoupledMWCAParams.
common_model: common_model of CoupledMWCAParams.
common_initial: common_initial of CoupledMWCAParams.
common_algorithms: common_algorithms of CoupledMWCAParams.
common_iteration: common_iteration of CoupledMWCAParams.
common_decomp: common_decomp of CoupledMWCAParams.
common_fix: common_fix of CoupledMWCAParams.
common_dims: common_dims of CoupledMWCAParams.
common_transpose: common_transpose of CoupledMWCAParams.
common_coretype: common_coretype of CoupledMWCAParams.
common_factors: Common factor matrices of CoupledMWCA.
common_cores: Common core tensors of CoupledMWCA.
specific_model: specific_model of CoupledMWCAParams.
specific_initial: specific_initial of CoupledMWCAParams.
specific_algorithms: specific_algorithms of CoupledMWCAParams.
specific_iteration: specific_iteration of CoupledMWCAParams.
specific_decomp: specific_decomp of CoupledMWCAParams.
specific_fix: specific_fix of CoupledMWCAParams.
specific_dims: specific_dims of CoupledMWCAParams.
specific_transpose: specific_transpose of CoupledMWCAParams.
specific_coretype: specific_coretype of CoupledMWCAParams.
specific_factors: Data specific factor matrices of CoupledMWCA.
specific_cores: Data specific core tensors of CoupledMWCA.
specific: specific of CoupledMWCAParams.
thr: thr of CoupledMWCAParams.
viz: viz of CoupledMWCAParams.
figdir: figdir of CoupledMWCAParams.
verbose: verbose of CoupledMWCAParams.
rec_error: The reconstructed error.
train_error: Training Error. train_error + test_error = rec_error.
test_error: Test Error. train_error + test_error = rec_error.
rel_change: The relative change of each iteration step.

See Also

[CoupledMWCAParams-class](#), [CoupledMWCA](#)

defaultCoupledMWCAParams

Default parameters for CoupledMWCA

Description

The input list is assumed to contain multiple arrays.

Usage

```
defaultCoupledMWCAParams(Xs, common_model)
```

Arguments

Xs	A list object containing multiple arrays
common_model	A list object to describe the relationship between dimensions of each tensor and factor matrices extracted from the tensor

Value

CoupledMWCAParams object.

Author(s)

Koki Tsuyuzaki

References

Andrzej Cichocki et al., (2016). Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions

Andrzej Cichocki et al., (2015). Tensor Decompositions for Signal Processing Applications, *IEEE SIGNAL PROCESSING MAGAZINE*

See Also

[CoupledMWCAParams-class](#), [CoupledMWCA](#), [checkCoupledMWCA](#), [initCoupledMWCA](#).

Examples

```

if(interactive()){
  # Test data (multiple arrays)
  Xs=list(
    X1=array(runif(7*4), dim=c(7,4)),
    X2=array(runif(4*5*6), dim=c(4,5,6)),
    X3=array(runif(6*8), dim=c(6,8)))
  # Setting of factor matrices
  common_model=list(
    X1=list(I1="A1", I2="A2"),
    X2=list(I2="A2", I3="A3", I4="A4"),
    X3=list(I4="A4", I5="A5"))
  # Default Parameters
  params <- defaultCoupledMWCAParams(Xs=Xs, common_model=common_model)
  # Perform Coupled MWCA
  out <- CoupledMWCA(params)
}

```

defaultMWCAParams	<i>Default parameters for MWCA</i>
-------------------	------------------------------------

Description

The input is assumed to be an array object.

Usage

```
defaultMWCAParams(X)
```

Arguments

X An array object

Value

MWCAParams object.

Author(s)

Koki Tsuyuzaki

References

Andrzej Cichocki et al., (2016). Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions

Andrzej Cichocki et al., (2015). Tensor Decompositions for Signal Processing Applications, *IEEE SIGNAL PROCESSING MAGAZINE*

See Also

[MWCAParams-class](#) and [MWCAResult-class](#).

Examples

```
if(interactive()){
  # Test data (single array)
  X <- nnTensor::toyModel("Tucker")@data
  # Default Parameters
  params <- defaultMWCAParams(X)
  # Perform MWCA
  out <- MWCA(params)
}
```

executeMWCAProgram	<i>Execute an MWCAProgram (Experimental)</i>
--------------------	--

Description

Compiles and runs an [MWCAProgram](#), including optional one-step factor refinements. For programs without refinements, this is equivalent to [compileMWCAProgram](#) followed by [CoupledMWCA](#) (or [MWCA](#)).

Refinements are applied after the main decomposition completes, using [refineFactor](#).

Usage

```
executeMWCAProgram(program, Xs, ...)
```

Arguments

<code>program</code>	An MWCAProgram object.
<code>Xs</code>	Named list of input arrays.
<code>...</code>	Additional parameters passed to compileMWCAProgram .

Value

A list with components:

fit The MWCAResult or CoupledMWCAResult.

refinements Named list of RefinedFactor objects (empty list if no refinements).

Author(s)

Koki Tsuyuzaki

See Also

[MWCAProgram](#), [compileMWCAProgram](#), [refineFactor](#).

initCoupledMWCA	<i>Initialize CoupledMWCA Factors and Cores</i>
-----------------	---

Description

Generates normalized, reproducible initial values for a CoupledMWCA problem without running iterative optimization. This is a companion to [checkCoupledMWCA](#) and can be used to inspect or store initial states before calling [CoupledMWCA](#).

Usage

```
initCoupledMWCA(params, seed = NULL, init_policy = "random")
```

Arguments

<code>params</code>	A CoupledMWCAParams object.
<code>seed</code>	An optional integer seed for reproducibility. When NULL (default), no seed management is performed.
<code>init_policy</code>	Character string specifying the initialization strategy. One of "random" (default), "svd", or "nonneg_random".

Value

A [CoupledMWCAInit](#) object with the following slots:

params The input CoupledMWCAParams.

common_factors Named list of initialized common factor matrices.

common_cores List of initialized common core tensors.

specific_factors Named list of initialized specific factor matrices (or `list(NULL)` when `specific=FALSE`).

specific_cores List of initialized specific core tensors (or `list(NULL)` when `specific=FALSE`).

init_policy The policy used.

seed The seed used (or NULL).

Author(s)

Koki Tsuyuzaki

References

Andrzej Cichocki et al., (2016). Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions

Andrzej Cichocki et al., (2015). Tensor Decompositions for Signal Processing Applications, *IEEE SIGNAL PROCESSING MAGAZINE*

See Also

[checkCoupledMWCA](#), [CoupledMWCA](#), [defaultCoupledMWCAParams](#).

Examples

```
if(interactive()){
  # Test data
  Xs <- toyModel("coupled_CP_Easy")
  common_model <- list(
    X1=list(I1="A1", I2="A2"),
    X2=list(I2="A2", I3="A3", I4="A4"),
    X3=list(I4="A4", I5="A5"))
  params <- defaultCoupledMWCAParams(Xs, common_model)
  # Initialize with seed for reproducibility
  init <- initCoupledMWCA(params, seed=42L)
  # Inspect shapes
  lapply(init@common_factors, dim)
  # SVD-based initialization
  init_svd <- initCoupledMWCA(params, seed=42L, init_policy="svd")
}
```

MWCA

Multi-way Component Analysis (MWCA)

Description

The input is assumed to be a MWCAParams object.

Usage

```
MWCA(params)
```

Arguments

params MWCAParams object

Value

MWCAResult object.

Author(s)

Koki Tsuyuzaki

References

Andrzej Cichocki et al., (2016). Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part I Low-Rank Tensor Decompositions
 Andrzej Cichocki et al., (2015). Tensor Decompositions for Signal Processing Applications, *IEEE SIGNAL PROCESSING MAGAZINE*

See Also

[MWCAParams-class](#) and [MWCAResult-class](#).

Examples

```
if(interactive()){
  # Test data (single array)
  X <- nnTensor::toyModel("Tucker")@data
  # Default Parameters
  params <- defaultMWCAParams(X)
  # Perform MWCA
  out <- MWCA(params)
}
```

MWCAParams-class	<i>Class "MWCAParams"</i>
------------------	---------------------------

Description

The parameter object to be specified against MWCA function.

Objects from the Class

Objects can be created by calls of the form `new("MWCAParams", ...)`.

Slots

X: A high-dimensional array.

mask: A mask array having the same dimension of X.

pseudocount: The pseudo count to avoid zero division, when the element is zero (Default: Machine Epsilon).

algorithms: Algorithms used to decompose the matricised tensor in each mode.

dims: The lower dimension of each factor matrix.

transpose: Whether the factor matrix is transposed to calculate core tensor.

viz: Whether the output is visualized.

figdir: When viz=TRUE, whether the plot is output in the directory.

Methods

MWCA Function to perform MWCA.

See Also

[MWCAResult-class](#), [MWCA](#)

Description

A lightweight S3-based internal representation for factorization programs. This is a structural description layer between problem specification and the existing solver parameters ([MWCAParams](#), [CoupledMWCAParams](#)).

Designed as a foundation for future recursive decomposition support (decomposing factors obtained from an initial decomposition). Currently only depth-0 (no refinement) and depth-1 (one-step factor refinement) programs are representable. Full recursive optimization is **not** implemented.

Usage

```
MWCAProgram(blocks, factors, refinements = list())
MWCAProgramBlock(modes, factor_map, type = "common", weight = 1)
MWCAProgramFactor(mode, dim, status = "decomposed",
  algorithm = "mySVD", type = "common")
MWCAProgramRefinement(source_factor, algorithm = "mySVD", dim = 2L)
```

Arguments

blocks	Named list of block descriptors (created by MWCAProgramBlock).
factors	Named list of factor descriptors (created by MWCAProgramFactor).
refinements	Named list of refinement descriptors (created by MWCAProgramRefinement). Default is empty.
modes	Character vector of mode names for the block.
factor_map	Named character vector mapping mode names to factor names.
type	One of "common" or "specific".
weight	Numeric weight for the block (default 1).
mode	Character. The mode name associated with a factor.
dim	Integer. Target lower dimension.
status	One of "decomposed" (updated by solver), "fixed" (initialized but not updated), or "frozen" (identity-like, not decomposed).
algorithm	Character or NULL. Decomposition algorithm name.
source_factor	Character. Name of the factor to refine.

Value

MWCAProgram returns an S3 object of class "MWCAProgram". MWCAProgramBlock, MWCAProgramFactor, and MWCAProgramRefinement return S3 objects of their respective classes.

Author(s)

Koki Tsuyuzaki

See Also

[validateMWCAProgram](#), [compileMWCAProgram](#), [CoupledMWCA](#), [MWCA](#).

Examples

```
if(interactive()){
  # A simple 3-block coupled program
  prog <- MWCAProgram(
    blocks=list(
      X1=MWCAProgramBlock(modes=c("I1","I2"),
        factor_map=c(I1="A1", I2="A2")),
      X2=MWCAProgramBlock(modes=c("I2","I3","I4"),
        factor_map=c(I2="A2", I3="A3", I4="A4")),
      X3=MWCAProgramBlock(modes=c("I4","I5"),
        factor_map=c(I4="A4", I5="A5"))),
    factors=list(
      A1=MWCAProgramFactor(mode="I1", dim=3),
      A2=MWCAProgramFactor(mode="I2", dim=3),
      A3=MWCAProgramFactor(mode="I3", dim=5),
      A4=MWCAProgramFactor(mode="I4", dim=4),
      A5=MWCAProgramFactor(mode="I5", dim=4)))
  print(prog)
  validateMWCAProgram(prog)

  # One-step refinement (structural only; compilation not yet supported)
  prog_ref <- MWCAProgram(
    blocks=prog$blocks, factors=prog$factors,
    refinements=list(
      R1=MWCAProgramRefinement(source_factor="A2",
        algorithm="myNMF", dim=2)))
  validateMWCAProgram(prog_ref)
}
```

MWCAResult-class	Class "MWCAResult"
------------------	--------------------

Description

The result object generated by MWCA function.

Slots

algorithms: algorithm of MWCAParams.

dims: dims of MWCAParams.

transpose: transpose of MWCAParams.

viz: viz of MWCAParams.

figdir: figdir of MWCAParams.

factors: The factor matrices of MWCA.

core: The core tensor of MWCA.

rec_error: The reconstructed error.

train_error: Training Error. $\text{train_error} + \text{test_error} = \text{rec_error}$.

test_error: Test Error. $\text{train_error} + \text{test_error} = \text{rec_error}$.

See Also

[MWCAParams-class](#), [MWCA](#)

myALS_SVD

*Alternating Least Square Singular Value Decomposition (ALS-SVD)
as an example of user-defined matrix decomposition.*

Description

The input data is assumed to be a matrix. When algorithms of MWCAParams and CoupledMWCA-Params are specified as "myALS_SVD", This function is called in MWCA and CoupledMWCA.

Usage

```
myALS_SVD(Xn, k, L2=1e-10, iter=30)
```

Arguments

Xn	The input matrix which has N-rows and M-columns.
k	The rank parameter ($k \leq \min(N, M)$)
L2	The regularization parameter (Default: 1e-10)
iter	The number of iteration (Default: 30)

Value

The output matrix which has N-rows and k-columns.

Author(s)

Koki Tsuyuzaki

References

Madeleine Udell et al., (2016). Generalized Low Rank Models, *Foundations and Trends in Machine Learning*, 9(1).

Examples

```

if(interactive()){
  # Test data
  matdata <- matrix(runif(10*20), nrow=10, ncol=20)
  # Perform ALS-SVD
  myALS_SVD(matdata, k=3, L2=0.1, iter=10)
}

```

myCX

CX Decomposition as an example of user-defined matrix decomposition.

Description

The input data is assumed to be a matrix. When algorithms of MWCAParams and CoupledMWCAParams are specified as "myCX", This function is called in MWCA and CoupledMWCA.

Usage

```
myCX(Xn, k)
```

Arguments

Xn The input matrix which has N-rows and M-columns.
 k The rank parameter ($k \leq \min(N, M)$)

Value

The output matrix which has N-rows and k-columns.

Author(s)

Koki Tsuyuzaki

References

Petros Drineas et al., (2008). Relative-Error CUR Matrix Decompositions, *SIAM Journal on Matrix Analysis and Applications*, 30(2), 844-881.

Examples

```

if(interactive()){
  # Test data
  matdata <- matrix(runif(10*20), nrow=10, ncol=20)
  # Perform CX
  myCX(matdata, k=3)
}

```

myICA	<i>Independent Component Analysis (ICA) as an example of user-defined matrix decomposition.</i>
-------	---

Description

The input data is assumed to be a matrix. When algorithms of MWCAParams and CoupledMWCAParams are specified as "myICA", This function is called in MWCA and CoupledMWCA.

Usage

```
myICA(Xn, k)
```

Arguments

Xn	The input matrix which has N-rows and M-columns.
k	The rank parameter ($k \leq \min(N,M)$)

Value

The output matrix which has N-rows and k-columns.

Author(s)

Koki Tsuyuzaki

References

A. Hyvarinen. (1999). Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks*, 10(3), 626-634.

Examples

```
if(interactive()){  
  # Test data  
  matdata <- matrix(runif(10*20), nrow=10, ncol=20)  
  # Perform ICA  
  myICA(matdata, k=3)  
}
```

myNMF	<i>Independent Component Analysis (ICA) as an example of user-defined matrix decomposition.</i>
-------	---

Description

The input data is assumed to be a matrix. When algorithms of MWCAParams and CoupledMWCAParams are specified as "myNMF", This function is called in MWCA and CoupledMWCA.

Usage

```
myNMF(Xn, k, L1=1e-10, L2=1e-10)
```

Arguments

Xn	The input matrix which has N-rows and M-columns.
k	The rank parameter ($k \leq \min(N, M)$)
L1	The regularization parameter to control the sparseness (Default: 1e-10)
L2	The regularization parameter to control the overfit (Default: 1e-10)

Value

The output matrix which has N-rows and k-columns.

Author(s)

Koki Tsuyuzaki

References

Andrzej CICHOCK, et. al., (2009). Nonnegative Matrix and Tensor Factorizations.

Examples

```
if(interactive()){
  # Test data
  matdata <- matrix(runif(10*20), nrow=10, ncol=20)
  # Perform NMF
  myNMF(matdata, k=3, L1=1e-1, L2=1e-2)
}
```

mySVD	<i>Singular Value Decomposition (SVD) as an example of user-defined matrix decomposition.</i>
-------	---

Description

The input data is assumed to be a matrix. When algorithms of MWCAParams and CoupledMWCAParams are specified as "mySVD", This function is called in MWCA and CoupledMWCA.

Usage

```
mySVD(Xn, k)
```

Arguments

Xn	The input matrix which has N-rows and M-columns.
k	The rank parameter ($k \leq \min(N, M)$)

Value

The output matrix which has N-rows and k-columns.

Author(s)

Koki Tsuyuzaki

Examples

```
if(interactive()){
  # Test data
  matdata <- matrix(runif(10*20), nrow=10, ncol=20)
  # Perform SVD
  mySVD(matdata, k=3)
}
```

plotTensor3Ds	<i>Plot function for visualization of tensor data structure</i>
---------------	---

Description

Multiple multi-dimensional arrays and matrices are visualized simultaneously.

Usage

```
plotTensor3Ds(Xs)
```

Arguments

Xs A List object containing multi-dimensional array (or matrix) in each element.

Author(s)

Koki Tsuyuzaki

See Also

[plotTensor3D](#) and [plotTensor2D](#).

Examples

```
Xs <- toyModel(model = "coupled_CP_Easy")

tmp <- tempdir()

png(filename=paste0(tmp, "/couled_CP.png"))
plotTensor3Ds(Xs)
dev.off()
```

RefinedFactor-class *Class "RefinedFactor"*

Description

Container for one-step factor refinement results produced by [refineFactor](#).

Given a factor matrix A ($k \times n$), the refinement decomposes $t(A)$ ($n \times k$) into $U * V$ where U is $n \times \text{dim}$ and V is $\text{dim} \times k$. The slots store:

- `sub_factors` = $t(U)$: the refined sub-basis ($\text{dim} \times n$)
- `coef` = $t(V)$: coefficients ($k \times \text{dim}$) such that A is approximately `coef %*% sub_factors`

Slots

source_object The original `MWCAResult` or `CoupledMWCAResult`.

source_factor_name Character. Name or index of the source factor.

source_factor Matrix. The original factor ($k \times n$).

algorithm Character. Algorithm used for refinement.

dim Integer. Target dimension.

sub_factors Matrix. Refined sub-basis ($\text{dim} \times n$).

coef Matrix. Coefficient matrix ($k \times \text{dim}$).

See Also

[refineFactor](#).

refineFactor

One-Step Factor Refinement (Experimental)

Description

Takes a factor matrix from a completed [MWCA](#) or [CoupledMWCA](#) fit and applies a single additional matrix factorization to decompose it further.

This is **not** a full recursive decomposition engine. Only one level of refinement is supported.

Usage

```
refineFactor(fit, factor_name, algorithm = "mySVD", dim = 2L)
```

Arguments

fit	An MWCAResult or CoupledMWCAResult object.
factor_name	For MWCAResult : an integer index. For CoupledMWCAResult : a character factor name (e.g. "A2").
algorithm	Character. Decomposition algorithm name.
dim	Integer. Target lower dimension for the refinement.

Details

Mathematical interpretation:

In [MWCA](#)/[CoupledMWCA](#), each factor matrix A has shape $k \times n$ where k is the lower dimension and n is the original observation dimension. Each row of A is a basis vector in the n -dimensional observation space.

`refineFactor` treats $t(A)$ ($n \times k$) as a new observed matrix and decomposes it: $t(A) = U * V$ where U is $n \times \text{dim}$ and V is $\text{dim} \times k$.

Equivalently, $A = t(V) * t(U)$, so the original k basis vectors are approximated by dim sub-basis vectors.

The returned `sub_factors` slot holds $t(U)$ ($\text{dim} \times n$): the new lower-rank basis vectors. The `coef` slot holds $t(V)$ ($k \times \text{dim}$): the coefficients expressing the original factor rows in terms of the sub-basis.

If the original decomposition was $X = S \times_m A_m \times \dots$, then after refinement of A_m : $X \sim (S \times_m \text{coef}) \times_m \text{sub_factors} \times \dots$. Computing this updated core is left to the caller.

Value

A [RefinedFactor](#) object with slots:

source_object The original fit object.

source_factor_name Character label identifying the source factor.

source_factor The original factor matrix A ($k \times n$).

algorithm Algorithm used.

dim Target dimension used.

sub_factors $t(U)$: the refined sub-basis (dim x n).

coef $t(V)$: coefficient matrix (k x dim) such that A is approximately `coef %*% sub_factors`.

Author(s)

Koki Tsuyuzaki

See Also

[MWCA](#), [CoupledMWCA](#), [RefinedFactor-class](#).

Examples

```
if(interactive()){
  X <- matrix(runif(20*30), nrow=20, ncol=30)
  params <- defaultMWCAParams(X)
  params@dims <- c(5L, 5L)
  fit <- MWCA(params)
  ref <- refineFactor(fit, 1L, algorithm="mySVD", dim=2L)
  dim(ref@sub_factors) # 2 x 20
  dim(ref@coef)        # 5 x 2
}
```

toyModel

Toy model of coupled tensor data

Description

A list object containing multiple arrays are generated.

Usage

```
toyModel(model = "coupled_CP_Easy", seeds=123)
```

Arguments

model	"coupled_CP_Easy", "coupled_CP_Hard", "coupled_Tucker_Easy", "coupled_Tucker_Hard", "coupled_Complex_Easy", or "coupled_Complex_Hard" can be specified (Default: "coupled_CP_Easy").
seeds	The seed of random number (Default: 123).

Author(s)

Koki Tsuyuzaki

Examples

```
Xs1 <- toyModel(model = "coupled_CP_Easy", seeds=123)
Xs2 <- toyModel(model = "coupled_CP_Hard", seeds=123)
Xs3 <- toyModel(model = "coupled_Tucker_Easy", seeds=123)
Xs4 <- toyModel(model = "coupled_Tucker_Hard", seeds=123)
Xs5 <- toyModel(model = "coupled_Complex_Easy", seeds=123)
Xs6 <- toyModel(model = "coupled_Complex_Hard", seeds=123)
```

validateMWCAProgram	<i>Validate an MWCAProgram</i>
---------------------	--------------------------------

Description

Checks structural consistency of a factorization program without running any computation. Validates factor references, mode mappings, and refinement depth constraints.

Usage

```
validateMWCAProgram(program)
```

Arguments

program An MWCAProgram object.

Value

A list with components:

ok Logical. TRUE if no errors.

errors Character vector of error messages.

warnings Character vector of warning messages.

summary Human-readable summary string.

Author(s)

Koki Tsuyuzaki

See Also

[MWCAProgram](#), [compileMWCAProgram](#).

Index

- * **classes**
 - CoupledMWCAInit-class, 6
 - MWCAProgram, 16
 - RefinedFactor-class, 23
 - * **methods**
 - checkCoupledMWCA, 3
 - compileMWCAProgram, 5
 - CoupledMWCA, 5
 - defaultCoupledMWCAParams, 10
 - defaultMWCAParams, 11
 - executeMWCAProgram, 12
 - initCoupledMWCA, 13
 - MWCA, 14
 - myALS_SVD, 18
 - myCX, 19
 - myICA, 20
 - myNMF, 21
 - mySVD, 22
 - plotTensor3Ds, 22
 - refineFactor, 24
 - toyModel, 25
 - validateMWCAProgram, 26
 - * **package**
 - mwTensor-package, 2
- checkCoupledMWCA, 3, 3, 10, 13, 14
- checkCoupledMWCA, CoupledMWCAParams-method (checkCoupledMWCA), 3
- compileMWCAProgram, 5, 12, 17, 26
- CoupledMWCA, 3, 4, 5, 5, 8, 10, 12–14, 17, 24, 25
- CoupledMWCA, CoupledMWCAInit-method (CoupledMWCA), 5
- CoupledMWCA, CoupledMWCAParams-method (CoupledMWCA), 5
- CoupledMWCAInit, 5, 13
- CoupledMWCAInit-class, 6
- CoupledMWCAParams, 16
- CoupledMWCAParams-class, 7
- CoupledMWCAResult-class, 9
- defaultCoupledMWCAParams, 4, 10, 14
- defaultMWCAParams, 11
- executeMWCAProgram, 12
- initCoupledMWCA, 3, 5–7, 10, 13
- initCoupledMWCA, CoupledMWCAParams-method (initCoupledMWCA), 13
- MWCA, 3, 5, 12, 14, 15, 17, 18, 24, 25
- MWCA, MWCAParams-method (MWCA), 14
- MWCAParams, 16
- MWCAParams-class, 15
- MWCAProgram, 3, 5, 12, 16, 26
- MWCAProgramBlock (MWCAProgram), 16
- MWCAProgramFactor (MWCAProgram), 16
- MWCAProgramRefinement (MWCAProgram), 16
- MWCAResult-class, 17
- mwTensor (mwTensor-package), 2
- mwTensor-package, 2
- myALS_SVD, 3, 18
- myCX, 3, 19
- myICA, 3, 20
- myNMF, 3, 21
- mySVD, 3, 22
- plotTensor2D, 23
- plotTensor3D, 23
- plotTensor3Ds, 3, 22
- print.MWCAProgram (MWCAProgram), 16
- RefinedFactor, 24
- RefinedFactor-class, 23
- refineFactor, 3, 12, 23, 24
- summary.MWCAProgram (MWCAProgram), 16
- toyModel, 25
- validateMWCAProgram, 5, 17, 26