Package 'CMTFtoolbox'

May 7, 2026

Title Create (Advanced) Coupled Matrix and Tensor Factorization Models

Version 1.0.1

Description Creation and selection of (Advanced) Coupled Matrix and Tensor Factorization (ACMTF) and ACMTF-Regression (ACMTF-R) models. Selection of the optimal number of components can be done using 'ACMTF_modelSelection()' and 'ACMTFR_modelSelection()'. The CMTF and ACMTF methods were originally described by Acar et al., 2011 <doi:10.48550/arXiv.1105.3422> and Acar et al., 2014 <doi:10.1186/1471-2105-15-239>, respectively.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports clue, doParallel, dplyr, foreach, ggplot2, magrittr, methods, mize, multiway, parallel, pracma, rTensor, stats, tidyr

Suggests knitr, ggpubr, NPLStoolbox, plotrix, parafac4microbiome, rmarkdown, testthat (>= 3.0.0), scales

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 2.10)

LazyData true

URL https://grvanderploeg.com/CMTFtoolbox/, https://github.com/GRvanderPloeg/CMTFtoolbox

BugReports https://github.com/GRvanderPloeg/CMTFtoolbox/issues

NeedsCompilation no

Author Geert Roelof van der Ploeg [aut, cre] (ORCID: <https://orcid.org/0009-0007-5204-3386>), Johan Westerhuis [ctb] (ORCID: <https://orcid.org/0000-0002-6747-9779>), Anna Heintz-Buschart [ctb] (ORCID: <https://orcid.org/0000-0002-9780-1933>), Age Smilde [ctb] (ORCID: <https://orcid.org/0000-0002-3052-4644>), University of Amsterdam [cph, fnd]

Maintainer Geert Roelof van der Ploeg <g.r.ploeg@uva.nl>

Repository CRAN

Date/Publication 2025-08-22 18:20:17 UTC

Contents

acmtfr_fg *Calculate function value of ACMTF*

Description

Calculate function value of ACMTF

Usage

```
acmtfr_fg(
  x,
  Z,
  Y,
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  pi = 0.5,
  mu = 1e-06
)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by setupCMTFdata().
Y	Dependent variable (regression part).
alpha	Alpha value of the loss function as specified by Acar et al., 2014
beta	Beta value of the loss function as specified by Acar et al., 2014
epsilon	Epsilon value of the loss function as specified by Acar et al., 2014
pi	Pi value of the loss function as specified by Van der Ploeg et al., 2025.
mu	Ridge term parameter for calculation of the regression coefficients rho (default = 1e-6).

Value

Scalar of the loss function value (when manual=FALSE), otherwise a list containing all loss terms.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
```

```
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)
Y = A[,1]

init = initializeACMTF(Z, 2, output="vect")
outcome = acmtfr_fg(init, Z, Y)
f = outcome$fn
g = outcome$gr
```

`acmtfr_fun` *Calculate function value of ACMTF*

Description

Calculate function value of ACMTF

Usage

```
acmtfr_fun(
  x,
  Z,
  Y,
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  pi = 0.5,
  mu = 1e-06,
  manual = FALSE
)
```

Arguments

`x`	Vectorized parameters of the CMTF model.
`Z`	Z object as generated by `setupCMTFdata()`.
`Y`	Dependent variable (regression part).
`alpha`	Alpha value of the loss function as specified by Acar et al., 2014
`beta`	Beta value of the loss function as specified by Acar et al., 2014
`epsilon`	Epsilon value of the loss function as specified by Acar et al., 2014
`pi`	Pi value of the loss function as specified by Van der Ploeg et al., 2025.
`mu`	Ridge term parameter for calculation of the regression coefficients rho (default = 1e-6).
`manual`	Manual calculation of each loss term (default FALSE).

Value

Scalar of the loss function value (when manual=FALSE), otherwise a list containing all loss terms.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)
Y = A[,1]

init = initializeACMTF(Z, 2, output="vect")
f = acmtfr_fun(init, Z, Y)
```

---

acmtfr_gradient *Calculate gradient of ACMTF model.*

Description

Calculate gradient of ACMTF model.

Usage

```
acmtfr_gradient(
  x,
  Z,
  Y,
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  pi = 0.5,
  mu = 1e-06
)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by `setupCMTFdata()`.
Y	Dependent variable (regression part).
alpha	Alpha value of the loss function as specified by Acar et al., 2014

beta	Beta value of the loss function as specified by Acar et al., 2014
epsilon	Epsilon value of the loss function as specified by Acar et al., 2014
pi	Pi value of the loss function as specified by Van der Ploeg et al., 2025.
mu	Ridge term parameter for calculation of the regression coefficients rho (default = 1e-6).

Value

Vectorized gradient of the ACMTF regression model.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)
Y = A[,1]

init = initializeACMTF(Z, 2, output="vect")
g = acmtfr_gradient(init, Z, Y)
```

ACMTFR_modelSelection *Model selection for ACMTFR*

Description

Model selection for ACMTFR

Usage

```
ACMTFR_modelSelection(
  datasets,
  modes,
  Y,
  sharedMode = 1,
  maxNumComponents = 5,
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  pi = 0.5,
  normalize = TRUE,
```

```
  normY = 1,
  method = "CG",
  cg_update = "HS",
  line_search = "MT",
  max_iter = 10000,
  max_fn = 10000,
  abs_tol = 1e-10,
  rel_tol = 1e-10,
  grad_tol = 1e-10,
  nstart = 5,
  numCores = 1,
  cvFolds = 2
)
```

Arguments

datasets	List of arrays of datasets. Multi-way and two-way may be combined.
modes	Numbered modes per dataset in a list. Example element 1: 1 2 3 and element 2: 1 4 for the X tensor and Y matrix case with a shared subject mode.
Y	Dependent variable (regression part).
sharedMode	Mode that is shared between all blocks, used to remove fibers for numFolds randomly initialized models.
maxNumComponents	Maximum number of components to check (default 3).
alpha	Scalar penalizing the components to be norm 1 (default 1).
beta	Vector of penalty values for each dataset, penalizing the lambda terms (default 1e-3).
epsilon	Scalar value to make it possible to compute the partial derivatives of lambda (default 1e-8).
pi	Pi value of the loss function as specified by Van der Ploeg et al., 2025.
normalize	Normalize the X blocks to frobenius norm 1 (default TRUE).
normY	Normalize Y to a specific value, (default: 1).
method	Optimization method to use (default = "CG", the conjugate gradient). See `mize::mize()` for other options.
cg_update	Update method for the conjugate gradient algorithm, see `mize::mize()` for the options (default="HS", Hestenes-Steifel).
line_search	Line search algorithm to use, see `mize::mize()` for the options (default="MT", More-Thuente).
max_iter	Maximum number of iterations.
max_fn	Maximum number of function evaluations.
abs_tol	Function tolerance criterion for convergence.
rel_tol	Relative function tolerance criterion for convergence.
grad_tol	Absolute tolerence for the l2-norm of the gradient vector.

nstart	Number of models to produce (default 1). If set higher than one, the package will return the best fitted model.
numCores	Number of cores to use (default 1). If set higher than one, the package will attempt to run in parallel.
cvFolds	Number of CV folds to create (default 10).

Value

List object containing plots of all metrics and dataframes containing the data used to create them.

Examples

```
set.seed(123)

I = 10
J = 5
K = 3
df = array(rnorm(I*J*K), c(I,J,K))
df2 = array(rnorm(I*J*K), c(I,J,K))
datasets = list(df, df2)
modes = list(c(1,2,3), c(1,4,5))
Y = as.matrix(rnorm(I))

# A very small procedure is run to limit computational requirements
result = ACMTFR_modelSelection(datasets,
                               modes,
                               Y,
                               pi=1.0,
                               maxNumComponents=2,
                               nstart=2,
                               cvFolds=2,
                               rel_tol=0.5,
                               abs_tol=0.5)

result$plots$overview
```

---

acmtfr_opt *Advanced coupled matrix and tensor factorizations*

Description

Advanced coupled matrix and tensor factorizations

Usage

```
acmtfr_opt(
  Z,
  Y,
  numComponents,
```

```
  initialization = "random",
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  pi = 0.5,
  mu = 1e-06,
  method = "CG",
  cg_update = "HS",
  line_search = "MT",
  max_iter = 10000,
  max_fn = 10000,
  abs_tol = 1e-10,
  rel_tol = 1e-10,
  grad_tol = 1e-10,
  nstart = 1,
  numCores = 1,
  sortComponents = TRUE,
  allOutput = FALSE
)
```

Arguments

Z	Combined dataset and mode object as produced by `setupCMTFdata()`.
Y	Dependent variable (regression part).
numComponents	Number of components
initialization	Initialization, either "random" (default) or "nvec" for numComponents components of the concatenated data using svd. Ignored and uses NPLS based initialization if pi=0.
alpha	Scalar penalizing the components to be norm 1 (default 1).
beta	Vector of penalty values for each dataset, penalizing the lambda terms (default 1e-3).
epsilon	Scalar value to make it possible to compute the partial derivatives of lambda (default 1e-8).
pi	Pi value of the loss function as specified by Van der Ploeg et al., 2025.
mu	Ridge term parameter for calculation of the regression coefficients rho (default = 1e-6).
method	Optimization method to use (default = "CG", the conjugate gradient). See `mize::mize()` for other options.
cg_update	Update method for the conjugate gradient algorithm, see `mize::mize()` for the options (default="HS", Hestenes-Steifel).
line_search	Line search algorithm to use, see `mize::mize()` for the options (default="MT", More-Thuente).
max_iter	Maximum number of iterations.
max_fn	Maximum number of function evaluations.
abs_tol	Function tolerance criterion for convergence.

rel_tol	Relative function tolerance criterion for convergence.
grad_tol	Absolute tolerence for the l2-norm of the gradient vector.
nstart	Number of models to produce (default 1). If set higher than one, the package will return the best fitted model.
numCores	Number of cores to use (default 1). If set higher than one, the package will attempt to run in parallel.
sortComponents	Sort the components in the output by descending order of variation explained.
allOutput	Return all created models. Ignored if nstart=1.

Value

List object, similar to `mize::mize()` output. Includes a Fac object of the model, which is a list of components per mode. Also includes an init object giving the initialized input vectors.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
Y = A[,1]
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

# specific setting to reduce runtime for CRAN
model = acmtfr_opt(Z, Y, 2, rel_tol=1e-5, abs_tol=1e-5)
```

---

acmtf_fg *Function value and gradient calculation for ACMTF*

Description

Function value and gradient calculation for ACMTF

Usage

```
acmtf_fg(x, Z, alpha = 1, beta = rep(0.001, length(Z$object)), epsilon = 1e-08)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by setupCMTFdata().
alpha	Alpha value of the loss function as specified by Acar et al., 2014
beta	Beta value of the loss function as specified by Acar et al., 2014
epsilon	Epsilon value of the loss function as specified by Acar et al., 2014

Value

A list containing the function ("fn") and the gradient ("gr").

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeACMTF(Z, 2, output="vect")
outcome = acmtf_fg(init, Z)
f = outcome$fn
g = outcome$gr
```

---

acmtf_fun *Calculate function value of ACMTF*

Description

Calculate function value of ACMTF

Usage

```
acmtf_fun(
  x,
  Z,
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  manual = FALSE
)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by setupCMTFdata().
alpha	Alpha value of the loss function as specified by Acar et al., 2014
beta	Beta value of the loss function as specified by Acar et al., 2014
epsilon	Epsilon value of the loss function as specified by Acar et al., 2014
manual	Manual calculation of each loss term (default FALSE)

Value

Scalar of the loss function value (when manual=FALSE), otherwise a list containing all loss terms

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeACMTF(Z, 2, output="vect")
f = acmtf_fun(init, Z)
```

---

acmtf_gradient *Calculate gradient of ACMTF model.*

Description

Calculate gradient of ACMTF model.

Usage

```
acmtf_gradient(
  x,
  Z,
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08
)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by setupCMTFdata().
alpha	Alpha value of the loss function as specified by Acar et al., 2014
beta	Beta value of the loss function as specified by Acar et al., 2014
epsilon	Epsilon value of the loss function as specified by Acar et al., 2014

Value

Vectorized gradient of the ACMTF model.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeACMTF(Z, 2, output="vect")
g = acmtf_gradient(init, Z)
```

ACMTF_modelSelection *Model selection for ACMTF*

Description

Model selection for ACMTF

Usage

```
ACMTF_modelSelection(
  datasets,
  modes,
  maxNumComponents = 3,
  sharedMode = 1,
  alpha = 1,
  beta = rep(0.001, length(datasets)),
  epsilon = 1e-08,
  nstart = 10,
  cvFolds = 10,
```

```
  numCores = 1,
  method = "CG",
  cg_update = "HS",
  line_search = "MT",
  max_iter = 10000,
  max_fn = 1e+05,
  rel_tol = 1e-08,
  abs_tol = 1e-08,
  grad_tol = 1e-08,
  plots = TRUE
)
```

Arguments

datasets	List of arrays of datasets. Multi-way and two-way may be combined.
modes	Numbered modes per dataset in a list. Example element 1: 1 2 3 and element 2: 1 4 for the X tensor and Y matrix case with a shared subject mode.
maxNumComponents	Maximum number of components to check (default 3).
sharedMode	Mode that is shared between all blocks, used to remove fibers for numFolds randomly initialized models.
alpha	Scalar penalizing the components to be norm 1 (default 1).
beta	Vector of penalty values for each dataset, penalizing the lambda terms (default 1e-3).
epsilon	Scalar value to make it possible to compute the partial derivatives of lambda (default 1e-8).
nstart	Number of randomly initialized models to create (default 10).
cvFolds	Number of CV folds to create (default 10).
numCores	Number of cores to use (default 1). A number higher than 1 will run the process in parallel.
method	Optimization method to use (default = "CG", the conjugate gradient). See `mize::mize()` for other options.
cg_update	Update method for the conjugate gradient algorithm, see `mize::mize()` for the options (default="HS", Hestenes-Steifel).
line_search	Line search algorithm to use, see `mize::mize()` for the options (default="MT", More-Thuente).
max_iter	Maximum number of iterations.
max_fn	Maximum number of function evaluations.
rel_tol	Relative function tolerance criterion for convergence.
abs_tol	Function tolerance criterion for convergence.
grad_tol	Absolute tolerence for the l2-norm of the gradient vector.
plots	Boolean to state if plots should be made of the outcome.

Value

List object containing variation explained, FMS, and degeneracy score metrics. If plots=TRUE, plots will be attached to the list.

Examples

```
set.seed(123)

I = 10
J = 5
K = 3
df = array(rnorm(I*J*K), c(I,J,K))
df2 = array(rnorm(I*J*K), c(I,J,K))
datasets = list(df, df2)
modes = list(c(1,2,3), c(1,4,5))

# A very small procedure is run to limit computational requirements
# Plots are not made to reduce CRAN runtime.
result = ACMTF_modelSelection(datasets,
                              modes,
                              maxNumComponents=1,
                              nstart=2,
                              cvFolds=2,
                              rel_tol=1e-1,
                              abs_tol=1e-1,
                              max_iter=2,
                              plots=FALSE)
```

---

acmtf_opt *Advanced coupled matrix and tensor factorizations*

Description

Advanced coupled matrix and tensor factorizations

Usage

```
acmtf_opt(
  Z,
  numComponents,
  initialization = "random",
  alpha = 1,
  beta = rep(0.001, length(Z$object)),
  epsilon = 1e-08,
  method = "CG",
  cg_update = "HS",
  line_search = "MT",
  max_iter = 10000,
```

```
  max_fn = 10000,
  abs_tol = 1e-10,
  rel_tol = 1e-10,
  grad_tol = 1e-10,
  nstart = 1,
  numCores = 1,
  sortComponents = TRUE,
  allOutput = FALSE
)
```

Arguments

Z	Combined dataset and mode object as produced by setupCMTFdata().
numComponents	Number of components
initialization	Initialization, either "random" (default) or "nvec" for numComponents components of the concatenated data using svd.
alpha	Scalar penalizing the components to be norm 1 (default 1).
beta	Vector of penalty values for each dataset, penalizing the lambda terms (default 1e-3).
epsilon	Scalar value to make it possible to compute the partial derivatives of lambda (default 1e-8).
method	Optimization method to use (default = "CG", the conjugate gradient). See mize::mize() for other options.
cg_update	Update method for the conjugate gradient algorithm, see mize::mize() for the options (default="HS", Hestenes-Steifel).
line_search	Line search algorithm to use, see mize::mize() for the options (default="MT", More-Thuente).
max_iter	Maximum number of iterations.
max_fn	Maximum number of function evaluations.
abs_tol	Function tolerance criterion for convergence.
rel_tol	Relative function tolerance criterion for convergence.
grad_tol	Absolute tolerence for the l2-norm of the gradient vector.
nstart	Number of models to produce (default 1). If set higher than one, the package will return the best fitted model.
numCores	Number of cores to use (default 1). If set higher than one, the package will attempt to run in parallel.
sortComponents	Sort the components in the output by descending order of variation explained.
allOutput	Return all created models. Ignored if nstart=1.

Value

List object, similar to mize::mize() output. Includes a Fac object of the model, which is a list of components per mode. Also includes an init object giving the initialized input vectors.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

# specific setting to reduce runtime for CRAN
model = acmtf_opt(Z, 1, rel_tol=1e-5, abs_tol=1e-5)
```

---

cmtf_fg *Function value and gradient calculation for CMTF*

Description

Function value and gradient calculation for CMTF

Usage

```
cmtf_fg(x, Z)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by `setupCMTFdata()`.

Value

A list containing the function ("fn") and the gradient ("gr").

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)
```

```
init = initializeCMTF(Z, 2, output="vect")
outcome = cmtf_fg(init, Z)
f = outcome$fn
g = outcome$gr
```

---

cmtf_fun *Calculate function value of CMTF*

Description

Calculate function value of CMTF

Usage

```
cmtf_fun(x, Z, manual = FALSE)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by setupCMTFdata().
manual	Manual calculation of each loss term (default FALSE)

Value

Function value of the CMTF loss value if manual=FALSE, otherwise a vector of the loss values per term.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeCMTF(Z, 2, output="vect")
f = cmtf_fun(init, Z)
```

cmtf_gradient *Calculate gradient of CMTF model.*

Description

Calculate gradient of CMTF model.

Usage

```
cmtf_gradient(x, Z)
```

Arguments

x	Vectorized parameters of the CMTF model.
Z	Z object as generated by `setupCMTFdata()`.

Value

Vectorized gradient of the CMTF model.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeCMTF(Z, 2, output="vect")
g = cmtf_gradient(init, Z)
```

cmtf_opt *Coupled matrix and tensor factorizations*

Description

Coupled matrix and tensor factorizations

Usage

```
cmtf_opt(
  Z,
  numComponents,
  initialization = "random",
  method = "CG",
  cg_update = "HS",
  line_search = "MT",
  max_iter = 10000,
  max_fn = 10000,
  abs_tol = 1e-08,
  rel_tol = 1e-08,
  grad_tol = 1e-08,
  nstart = 1,
  numCores = 1,
  sortComponents = TRUE,
  allOutput = FALSE
)
```

Arguments

Z	Combined dataset and mode object as produced by `setupCMTFdata()`.
numComponents	Number of components
initialization	Initialization, either "random" (default) or "nvec" for numComponents components of the concatenated data using svd.
method	Optimization method to use (default = "CG", the conjugate gradient). See `mize::mize()` for other options.
cg_update	Update method for the conjugate gradient algorithm, see `mize::mize()` for the options (default="HS", Hestenes-Steifel).
line_search	Line search algorithm to use, see `mize::mize()` for the options (default="MT", More-Thuente).
max_iter	Maximum number of iterations.
max_fn	Maximum number of function evaluations.
abs_tol	Function tolerance criterion for convergence.
rel_tol	Relative function tolerance criterion for convergence.
grad_tol	Absolute tolerence for the l2-norm of the gradient vector.
nstart	Number of models to produce (default 1). If set higher than one, the package will return the best fitted model.
numCores	Number of cores to use (default 1). If set higher than one, the package will attempt to run in parallel.
sortComponents	Sort the components in the output by descending order of variation explained.
allOutput	Return all created models. Ignored if nstart=1.

Value

List object, similar to `mize::mize()` output. Includes a Fac object of the model, which is a list of components per mode. Also includes an init object giving the initialized input vectors.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

model = cmtf_opt(Z, 1, rel_tol=1e-4) # quick convergence for example only
```

---

computeFMS *Compute Factor Match Score for two models.*

Description

Compute Factor Match Score for two models.

Usage

```
computeFMS(Fac1, Fac2, modes)
```

Arguments

Fac1	A list of matrices corresponding to found components per mode in model 1.
Fac2	A list of matrices corresponding to found components per mode in model 2.
modes	List of modes per dataset.

Value

Vector of FMS scores, one per dataset.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100, 2))
E = array(rnorm(10*2), c(10, 2))
```

```
Fac1 = list(A,B,C,D,E)
Fac2 = Fac1 # identical models for the purposes of demonstration
modes = list(c(1,2,3), c(1,4,5))
FMS_result = computeFMS(Fac1, Fac2, modes) # FMS_result = c(1,1)
```

`degenScore` *degenScore*

Description

Computes the maximum absolute off-diagonal Tucker congruence coefficient between subject-mode components in an ACMTF model. This metric serves as a diagnostic tool to detect potential degeneracy in the subject-mode loadings.

Usage

```
degenScore(A)
```

Arguments

`A`	A numeric matrix of subject-mode loadings (dimensions: subjects x components).

Details

A high `degenScore` (e.g., > 0.85) indicates that two or more components in the subject mode are highly similar, suggesting a possible degeneracy or lack of uniqueness. A low value (e.g., < 0.3) indicates well-separated components.

Value

A numeric scalar representing the maximum absolute off-diagonal Tucker congruence between components.

Examples

```
# Example: Compute degenScore for a random loading matrix
A <- matrix(rnorm(100), nrow = 10, ncol = 10)
degenScore(A)
```

fac_to_vect *Vectorize Fac object*

Description

Vectorize Fac object

Usage

```
fac_to_vect(Fac)
```

Arguments

Fac	Fac object from CMTF and ACMTF

Value

Vectorized Fac object

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))
Fac = list(A, B, C, D, E)
v = fac_to_vect(Fac)
```

FMS_cv *Compute Factor Match Score for two models.*

Description

Compute Factor Match Score for two models.

Usage

```
FMS_cv(Fac1, Fac2, sharedMode = 1)
```

Arguments

Fac1	A list of matrices corresponding to found components per mode in model 1.
Fac2	A list of matrices corresponding to found components per mode in model 2.
sharedMode	The shared mode that is excluded from FMS calculation.

Value

Scalar of FMS value

Examples

```
set.seed(123)

I = 10
J = 5
K = 3
df = array(rnorm(I*J*K), c(I,J,K))
datasets = list(df, df)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes)

model1 = acmtf_opt(Z, 1)

Fac1 = model1$Fac[1:3]
Fac2 = Fac1 # identical models for the purposes of demonstration
result = FMS_cv(Fac1, Fac2) # [1] 1
```

FMS_random *Compute Factor Match Score for two models.*

Description

Compute Factor Match Score for two models.

Usage

```
FMS_random(Fac1, Fac2)
```

Arguments

Fac1	A list of matrices corresponding to found components per mode in model 1.
Fac2	A list of matrices corresponding to found components per mode in model 2.

Value

Scalar of FMS value

Examples

```
set.seed(123)

I = 10
J = 5
K = 3
df = array(rnorm(I*J*K), c(I,J,K))
```

```
datasets = list(df, df)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes)

model1 = acmtf_opt(Z, 1)

Fac1 = model1$Fac[1:3]
Fac2 = Fac1 # identical models for the purposes of demonstration
result = FMS_random(Fac1, Fac2) # [1] 1
```

`Georgiou2025` *Georgiou2025 Apical Periodontitis data*

Description

The Georgiou longitudinal dataset as a three-dimensional array and a matrix, with subjects in mode 1, features in mode 2, and time in mode3.

Usage

```
Georgiou2025
```

Format

`Georgiou2025`**:**
A list object with two elements

Inflammatory_mediators Longitudinally measured inflammatory mediator data.

Tooth_microbiome Single-timepoint tooth microbiome data of the extracted tooth

Source

TBD

`initializeACMTF` *Initialize input vectors for the ACMTF algorithm*

Description

Initialize input vectors for the ACMTF algorithm

Usage

```
initializeACMTF(
  Z,
  numComponents,
  initialization = "random",
  output = "Fac",
  Y = NULL
)
```

Arguments

Z	List object as generated by setupCMTFdata().
numComponents	Integer stating the number of desired components for the CMTF model.
initialization	Initialization method, either "random" or "nvec" (default "random"). Random will initialize random input vectors. Nvec will initialize vectors according to an singular value decomposition of the (matricized, if needed) concatenated datasets per mode.
output	How to return output: as a "Fac" object (default) or vectorized ("vect").
Y	Used as dependent variable when initialization is set to "npls". Not used by default.

Value

List or vector of initialized input vectors per mode.

Examples

```
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeACMTF(Z, 2)
```

---

initializeCMTF *Initialize input vectors for the CMTF algorithm*

Description

Initialize input vectors for the CMTF algorithm

Usage

```
initializeCMTF(
  Z,
  numComponents,
  initialization = "random",
  output = "Fac",
  Y = NULL
)
```

Arguments

Z	List object as generated by setupCMTFdata().
numComponents	Integer stating the number of desired components for the CMTF model.
initialization	Initialization method, either "random" or "nvec" (default "random"). Random will initialize random input vectors. Nvec will initialize vectors according to an singular value decomposition of the (matricized, if needed) concatenated datasets per mode.
output	How to return output: as a "Fac" object (default) or vectorized ("vect").
Y	Used as dependent variable when initialization is set to "npls". Not used by default.

Value

List or vector of initialized input vectors per mode.

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

init = initializeCMTF(Z, 1)
```

---

normalizeFac *Normalize all vectors in model output Fac object to norm 1.*

Description

Normalize all vectors in model output Fac object to norm 1.

Usage

```
normalizeFac(Fac, modes)
```

Arguments

Fac	List object with all components per mode per item.
modes	List object with modes per dataset (see also setupCMTFdata())

Value

List object of normalized Fac object, the extracted norms per loading vector per component, and the norms per dataset per component.

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))
modes = list(c(1,2,3), c(1,4,5))

Fac = list(A, B, C, D, E)
output = normalizeFac(Fac, modes)
```

---

npred *Predict Y for new data by projecting the data onto the latent space defined by an ACMTF-R model.*

Description

Predict Y for new data by projecting the data onto the latent space defined by an ACMTF-R model.

Usage

```
npred(model, newX, Z, sharedMode = 1)
```

Arguments

model	ACMTF-R model
newX	List object of new data, where each element corresponds to a block
Z	Original input data used for the model
sharedMode	Shared mode between the blocks (default 1).

Value

Ypred: the predicted value of Y for the new data

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100, 2))
E = array(rnorm(10*2), c(10, 2))
```

```
df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes)
Y = matrix(A[,1])
# Remove a sample and define
i = 1
Xtest = lapply(Z$object, function(x){x@data[i,,]})
Ytest = Y[i]
Xtrain = lapply(Z$object, function(x){x@data[-i,,]})
Ytrain = Y[-i]
Ztrain = setupCMTFdata(Xtrain, Z$modes)
model = acmtfr_opt(Ztrain,Ytrain,1,initialization="random",pi=1, nstart=1, max_iter=10)
Ypred = npred(model, Xtest, Ztrain, sharedMode=1)
```

`reinflateFac` *Reinflate all datablocks from a model Fac object.*

Description

Basically a wrapper function for `reinflateTensor()` and `reinflateMatrix()`.

Usage

```
reinflateFac(Fac, Z, returnAsTensor = FALSE)
```

Arguments

`Fac`	Fac object output from CMTF and ACMTF
`Z`	Z object as generated by `setupCMTFdata()`.
`returnAsTensor`	Boolean to return data blocks as rTensor tensor objects (default FALSE)

Value

List of data blocks

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
```

```
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

result = cmtf_opt(Z, 1, max_iter=2)
Xhats = reinflateFac(result$Fac, Z)
```

`reinflateMatrix` *Create a matrix from a matrix of scores and loadings similar to a component model.*

Description

Create a matrix from a matrix of scores and loadings similar to a component model.

Usage

```
reinflateMatrix(A, B)
```

Arguments

A	I x N matrix corresponding to scores for N components.
B	J x N matrix corresponding to loadings for N components.

Value

M, an I x J matrix.

Examples

```
A = rnorm(108)
B = rnorm(100)
M = reinflateMatrix(A,B)
```

`reinflateTensor` *Create a tensor out of a set of matrices similar to a component model.*

Description

Create a tensor out of a set of matrices similar to a component model.

Usage

```
reinflateTensor(A, B, C)
```

Arguments

A	I x N matrix corresponding to loadings in the first mode for N components.
B	J x N matrix corresponding to loadings in the second mode for N components.
C	K x N matrix corresponding to loadings in the third mode for N components.

Value

M, an I x J x K tensor.

Examples

```
A = rnorm(108)
B = rnorm(100)
C = rnorm(10)
M = reinflateTensor(A,B,C)
```

---

removeTwoNormCol *Remove two-norms column-wise from a matrix*

Description

Remove two-norms column-wise from a matrix

Usage

```
removeTwoNormCol(df)
```

Arguments

df	Matrix of loadings

Value

Matrix of loadings where the column-wise 2-norm is 1.

Examples

```
A = array(rnorm(108*4), c(108,4))
Anorm = removeTwoNormCol(A)
```

`setupCMTFdata` *Set up datasets for (A)CMTF input*

Description

Set up datasets for (A)CMTF input

Usage

```
setupCMTFdata(datasets, modes, normalize = TRUE)
```

Arguments

`datasets`	List of arrays of datasets. Multi-way and two-way may be combined.
`modes`	Numbered modes per dataset in a list. Example element 1: 1 2 3 and element 2: 1 4 for the X tensor and Y matrix case with a shared subject mode.
`normalize`	Boolean specifying if the datasets should be normalized to Frobenium norm 1. Note: this function puts zeroes in positions with missing values. The indices of missing data are conserved in the output.

Value

Z, a list with "object" listing the datasets, "sizes" with their size, "norms" with their norms and "missing" stating the missing data.

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)
```

vect_to_fac *Convert vectorized output of (a)cmtf to a Fac list object with all loadings per mode.*

Description

Convert vectorized output of (a)cmtf to a Fac list object with all loadings per mode.

Usage

```
vect_to_fac(vect, Z, sortComponents = FALSE)
```

Arguments

vect	Vectorized output of (a)cmtf
Z	Original Z input object (see setupCMTFdata).
sortComponents	Sort the order of the components by variation explained (default FALSE).

Value

Fac: list object with all loadings in all components per mode, ordered the same way as Z$modes.

Examples

```
set.seed(123)
A = array(rnorm(108*2), c(108, 2))
B = array(rnorm(100*2), c(100, 2))
C = array(rnorm(10*2), c(10, 2))
D = array(rnorm(100*2), c(100,2))
E = array(rnorm(10*2), c(10,2))

df1 = reinflateTensor(A, B, C)
df2 = reinflateTensor(A, D, E)
datasets = list(df1, df2)
modes = list(c(1,2,3), c(1,4,5))
Z = setupCMTFdata(datasets, modes, normalize=FALSE)

result = cmtf_opt(Z, 2, initialization="random", max_iter = 2)
Fac = vect_to_fac(result$par, Z)
```

Index