Package 'tensorMiss'

May 8, 2026

Type Package

Title Handle Missing Tensor Data with C++ Integration

Version 1.1.1

Date 2024-04-09

Author Zetai Cen [aut, cre]

Maintainer Zetai Cen <z.cen@lse.ac.uk>

Description To handle higher-order tensor data. See Kolda and Bader (2009) <doi:10.1137/07070111X> for details on tensor. While existing packages on tensor data extend the base 'array' class to some data classes, this package serves as an alternative resort to handle tensor only as 'array' class.
Some functionalities related to missingness are also supported.

License GPL-3

Imports Rcpp (>= 1.0.11), RcppEigen, rTensor, stats

Encoding UTF-8

LinkingTo Rcpp, RcppEigen

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-04-09 22:10:02 UTC

Contents

`fle` *Factor loading error*

Description

Computing the column space distance between two matrix

Usage

```
fle(A1, A2)
```

Arguments

`A1`	A matrix of m rows and n columns.
`A2`	A matrix of m rows and l columns where l can equal n.

Value

A numeric number

Examples

```
fle(matrix(1:12, nrow=4), matrix(11:22, nrow=4));
```

`miss_factor_est` *Estimation of tensor factor models with missing data*

Description

Estimate the factor structure on an order-K tensor at each time t, with maximum K as 3 and missing entries allowed

Usage

```
miss_factor_est(dt, r = 0, delta = 0.2)
```

Arguments

dt	Tensor time series, written in an array with dimension K+1 and mode-1 as the time mode.
r	Rank of core tensors, written in a vector of length K. First value as 0 is to denote unknown rank which would be automatically estimated using ratio-based estimators. Default is 0.
delta	Non-negative number as the correction parameter for rank estimation. Default is 0.2.

Value

A list containing the following: r: a vector representing either the given rank or the estimated rank, with length K; A: a list of estimated K factor loading matrices; Ft: the estimated core factor series, as multi-dimensional array with dimension K+1, where mode-1 is the time mode; imputation: the imputed common component time series, as multi-dimensional array with dimension K+1, where mode-1 is the time mode; covMatrix: a list of estimated covariance matrix which are used to estimate loading matrices;

Examples

```
K = 3;
TT = 10;
d = c(20,20,20);
r = c(2,2,2);
re = c(2,2,2);
eta = list(c(0,0), c(0,0), c(0,0));
coef_f = c(0.7, 0.3, -0.4, 0.2, -0.1);
coef_fe = c(-0.7, -0.3, -0.4, 0.2, 0.1);
coef_e = c(0.8, 0.4, -0.4, 0.2, -0.1);
data_test = tensor_gen(K,TT,d,r,re,eta, coef_f, coef_fe, coef_e);
data_miss = miss_gen(data_test$X);
miss_factor_est(data_miss, r);
```

---

miss_gen *Assignment of missingness to tensor time series*

Description

Assign missingness to a given order-K tensor time series, where the maximum K is 4

Usage

```
miss_gen(dt, type = "random", p = 0.7)
```

Arguments

dt	Tensor time series, written in an array with dimension K+1 and mode-1 as the time mode.
type	Type of missingness, where "random" is random missing with probability p, "simul" is missingness on the last half along all dimensions, "mix" is a mixture of "random" and "simul". Default is "random".
p	If type is "random", then each entry is randomly missing with probability 1-p. Default is 0.7.

Value

A multi-dimensional array with dimension K+1, where mode-1 is the time mode and missing entries are denoted by NA

Examples

```
K = 3;
TT = 10;
d = c(20,20,20);
r = c(2,2,2);
re = c(2,2,2);
eta = list(c(0,0), c(0,0), c(0,0));
coef_f = c(0.7, 0.3, -0.4, 0.2, -0.1);
coef_fe = c(-0.7, -0.3, -0.4, 0.2, 0.1);
coef_e = c(0.8, 0.4, -0.4, 0.2, -0.1);
tensor_gen(K,TT,d,r,re,eta, coef_f, coef_fe, coef_e);
data_test = tensor_gen(K,TT,d,r,re,eta, coef_f, coef_fe, coef_e);
miss_gen(data_test$X);
```

---

qMSE *Quantile relative squared error*

Description

Computing the q-quantile relative squared error as a generalised error measure on relative mean squared error

Usage

```
qMSE(x_true, x_est, q = 100)
```

Arguments

x_true	True values, written in a vector of length n.
x_est	Imputed or estimated values, written in a vector of length n.
q	Number of partition intervals. If q equals n, then output is essentially relative mean squared error. Default is 100.

Value

A numeric number

Examples

```
qMSE(c(2, 3, 7, 1), c(-2, 0.5, 8, 2), 1);
```

`refold` *Tensor refolding*

Description

Performing to matrices tensorisation, which is the inverse process of unfolding

Usage

```
refold(unfolding, k, dim_vec)
```

Arguments

`unfolding`	A matrix.
`k`	An integer specifying the mode of array to refold from.
`dim_vec`	A vector specifying the expected dimension of output array.

Value

A multi-dimensional array

Examples

```
refold(matrix(1:9,nrow=3), 1, c(3,1,3));
```

sigmaD *HAC covariance estimator for asymptotic normality on each row j of loading matrix estimator*

Description

Computing the HAC covariance estimator for asymptotic normality on each row j of the mode-k loading matrix estimator, with maximum order of tensor time series as 3

Usage

```
sigmaD(k, D, Q, C, Y, j, beta = 0)
```

Arguments

k	Mode of loading matrix.
D	Eigenvalue matrix of sample covariance matrix, with dimension rk by rk.
Q	Estimated mode-k loading matrix, with dimension Ik by rk.
C	Estimated common component series, written in an array with dimension K+1 and mode-1 as the time mode.
Y	Observed time series with missingness allowed, written in an array with dimension K+1 and mode-1 as the time mode.
j	Integer representing the row of mode-k loading matrix. Value should be integers from minimum 1 to maximum Ik.
beta	Lag parameter of the HAC type. Default is 0.

Value

A matrix of dimension rk by rk

Examples

```
K = 3;
TT = 10;
d = c(20,20,20);
r = c(2,2,2);
re = c(2,2,2);
eta = list(c(0,0), c(0,0), c(0,0));
coef_f = c(0.7, 0.3, -0.4, 0.2, -0.1);
coef_fe = c(-0.7, -0.3, -0.4, 0.2, 0.1);
coef_e = c(0.8, 0.4, -0.4, 0.2, -0.1);
data_test = tensor_gen(K,TT,d,r,re,eta, coef_f, coef_fe, coef_e);
data_miss = miss_gen(data_test$X);
data_est = miss_factor_est(data_miss, r);
D = diag(x=(svd(data_est$covMatrix[[2]])$d)[1:2], nrow=2, ncol=2);
sigmaD(2, D, data_est$A[[2]], data_est$imputation, data_miss, 2, 2);
```

tensor_gen *Data generation of tensor time series with factor structure*

Description

Generate an order-K tensor at each time t, with the first mode as the time mode and maximum allowed K is 4

Usage

```
tensor_gen(
  K,
  TT,
  d,
  r,
  re,
  eta,
  coef_f,
  coef_fe,
  coef_e,
  heavy_tailed = FALSE,
  t_df = 3,
  seed = 2023
)
```

Arguments

K	Order of the generated tensor at each time t.
TT	Length of time series.
d	Dimensions of each mode of the tensor, written in a vector of length K.
r	Rank of the core tensors, written in a vector of length K.
re	Rank of the cross-sectional common error core tensors, written in a vector of length K.
eta	Quantities controlling factor strengths in each factor loading matrix, written in a list of K vectors.
coef_f	AR(5) coefficients for the factor series, written in a vector of length 5.
coef_fe	AR(5) coefficients for the common component in error series, written in a vector of length 5.
coef_e	AR(5) coefficients for the idiosyncratic component in error series, written in a vector of length 5.
heavy_tailed	Whether to generate data from heavy-tailed distribution. If FALSE, generate from N(0,1); if TRUE, generate from t-distribution. Default is FALSE.
t_df	The degree of freedom for t-distribution if heavy_tailed = TRUE. Default is 3.
seed	Random seed required for reproducibility. Default is 2023.

Value

A list containing the following: X: the generated tensor time series, as multi-dimensional array with dimension K+1, where mode-1 is the time mode; A: a list of K factor loading matrices; C: the generated common component time series, as multi-dimensional array with dimension K+1, where mode-1 is the time mode; Ft: the generated core factor series, as multi-dimensional array with dimension K+1, where mode-1 is the time mode;

Examples

```
K = 3;
TT = 10;
d = c(20,20,20);
r = c(2,2,2);
re = c(2,2,2);
eta = list(c(0,0), c(0,0), c(0,0));
coef_f = c(0.7, 0.3, -0.4, 0.2, -0.1);
coef_fe = c(-0.7, -0.3, -0.4, 0.2, 0.1);
coef_e = c(0.8, 0.4, -0.4, 0.2, -0.1);
tensor_gen(K,TT,d,r,re,eta, coef_f, coef_fe, coef_e);
```

---

ttm *Mode k product with matrix*

Description

Performing k-mode matrix product of a tensor to a matrix

Usage

```
ttm(ten, A, k)
```

Arguments

ten	A multi-dimensional array with the mode-k dimension m.
A	A matrix with dimension n by m.
k	An integer specifying the tensor mode to perform k-mode matrix product.

Value

A multi-dimensional array with the k mode dimension n

Examples

```
ttm(array(1:24,c(3,4,2)), matrix(1:4,nrow =2), 3);
```

`unfold` *Tensor unfolding*

Description

Performing to multi-dimensional arrays tensor unfolding, also known as matricization

Usage

```
unfold(ten, k)
```

Arguments

`ten`	A multi-dimensional array.
`k`	An integer specifying the mode of array to unfold.

Value

A matrix

Examples

```
unfold(array(1:24, dim=c(3,4,2)), 2);
```

Index