Package: CoFM (via r-universe)

May 28, 2026

Type Package

Title Copula Factor Models

Version 1.1.4

Author Guangbao Guo [aut, cre], Xin Gao [aut]

Maintainer Guangbao Guo <ggb11111111@163.com>

Description Provides tools for factor analysis in high-dimensional settings under copula-based factor models. It includes functions to simulate factor-model data with copula-distributed idiosyncratic errors (e.g., Clayton, Gumbel, Frank, Student t and Gaussian copulas) and to perform diagnostic tests such as the Kaiser-Meyer-Olkin measure and Bartlett's test of sphericity. Estimation routines include principal component based factor analysis, projected principal component analysis, and principal orthogonal complement thresholding for large covariance matrix estimation. The philosophy of the package is described in Guo G. (2023) <doi:10.1007/s00180-022-01270-z>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0), spelling

NeedsCompilation no

Imports MASS, psych, copula, matrixcalc, stats

Language en-US

Config/pak/sysreqs libgsl0-dev

Repository https://guangbaog.r-universe.dev

Date/Publication 2026-01-27 20:50:07 UTC

RemoteUrl https://github.com/cran/CoFM

RemoteRef HEAD

RemoteSha 81e63bb21d55d2eed59a6e37f7e1e78f54224681

Contents

`air_quality` *Air Quality Data Set*

Description

Air quality measurements collected from a gas multisensor device deployed in an Italian city.

This dataset contains the responses of a gas multisensor device deployed on the field in an Italian city.

Usage

```
air_quality

air_quality
```

Format

A data frame with 9358 rows and 14 variables:

`DateTime` Date and time of the measurement (`POSIXct`).

`CO_GT` True hourly averaged CO concentration in mg/m^3 (reference analyzer).

`PT08_S1_CO` PT08.S1 (tin oxide) hourly averaged sensor response (CO targeted).

`NMHC_GT` True hourly averaged non-methanic hydrocarbons concentration in $\mu g/m^3$.

`C6H6_GT` True hourly averaged benzene concentration in $\mu g/m^3$.

`PT08_S2_NMHC` PT08.S2 (titania) hourly averaged sensor response (NMHC targeted).

`NOx_GT` True hourly averaged NOx concentration in ppb.

`PT08_S3_NOx` PT08.S3 (tungsten oxide) hourly averaged sensor response (NOx targeted).

`NO2_GT` True hourly averaged NO2 concentration in $\mu g/m^3$.

`PT08_S4_NO2` PT08.S4 (tungsten oxide) hourly averaged sensor response (NO2 targeted).

PT08_S5_O3 PT08.S5 (indium oxide) hourly averaged sensor response (O3 targeted).

T Temperature in degrees Celsius.

RH Relative humidity (percent).

AH Absolute humidity.

A data frame with 9357 rows and 16 variables.

Details

The dataset was collected from March 2004 to February 2005. Missing values are tagged with -200 value in the raw data, but have been converted to NA.

Source

UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Air+Quality
https://archive.ics.uci.edu/ml/datasets/Air+Quality

CoFM *Generate Copula Factor Models Data and Perform Tests*

Description

This function simulates data based on a Copula Factor Model structure. It generates factor scores, factor loadings, and error terms (using specified Copula distributions), combines them to create the observed data, and then performs KMO and Bartlett's tests on the generated data.

Usage

```
CoFM(n = 1000, p = 10, m = 5, type = "Clayton", param = 2)
```

Arguments

n	Integer. Sample size (number of rows). Default is 1000.
p	Integer. Number of observed variables (columns). Default is 10.
m	Integer. Number of factors. Default is 5.
type	Character. The type of Copula for error terms. Options: "Clayton", "Gumbel", "Frank".
param	Numeric. The parameter for the Copula (theta). Default is 2.0.

Value

A list containing:

data	The generated data matrix (n x p).
KMO	The results of the Kaiser-Meyer-Olkin test.
Bartlett	The results of Bartlett's test of sphericity.
True_Params	A list containing the true parameters used (F, A, D, mu).

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# Clayton copula errors (toy size)
res1 <- CoFM::CoFM(n = 200, p = 6, m = 2, type = "Clayton", param = 2)
res1$KMO
res1$Bartlett

# Gumbel copula errors (toy size)
res2 <- CoFM::CoFM(n = 150, p = 6, m = 2, type = "Gumbel", param = 2)
head(res2$data)
```

`Copula_errors` *Generate Copula-Distributed Error Terms*

Description

Generate random samples (error terms) from various copula distributions, including Archimedean (Clayton, Gumbel, Frank), Elliptical (t, Normal), Mixed, and Extreme-Value (Galambos) copulas. Useful for simulation studies involving non-normal error structures.

Usage

```
Copula_errors(
  n,
  type = "Clayton",
  dim = 2,
  param = NULL,
  extra_params = list()
)
```

Arguments

`n`	Integer. The number of samples (rows) to generate.
`type`	Character. The type of Copula to use. Options: "Clayton", "Gumbel", "Frank", "t", "Mixed", "Galambos", "Normal".
`dim`	Integer. The dimension of the copula (number of columns/variables). Used for Archimedean/Elliptical copulas. For "Mixed" the default is 3.
`param`	Numeric or Matrix. The main parameter for the copula (e.g., theta for Archimedean, correlation vector/matrix for Normal). If NULL, default values are used.
`extra_params`	List. Additional parameters for specific copulas (e.g., `df` for t-Copula, `w` for Mixed).

Value

A numeric matrix of dimension (n x dim) containing the generated random samples.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# Example 1: Clayton Copula (toy example)
U_clayton <- Copula_errors(n = 200, type = "Clayton", dim = 2, param = 2)
head(U_clayton)

# Example 2: t-Copula with degrees of freedom (toy example)
U_t <- Copula_errors(
  n = 200, type = "t", dim = 2, param = 0.7,
  extra_params = list(df = 4)
)
head(U_t)

# Example 3: Multivariate Normal Copula (dim = 3)
# normalCopula() expects the upper-triangular correlations as a vector:
# (rho_12, rho_13, rho_23) for dim=3
rho_vec <- c(0.5, 0.3, 0.4)
U_normal <- Copula_errors(n = 200, type = "Normal", dim = 3, param = rho_vec)
head(U_normal)
```

FanPC_basic *Perform Basic FanPC Factor Analysis*

Description

This function performs factor analysis using a principal-component (FanPC) approach. It estimates the factor loading matrix and uniquenesses from the correlation matrix of the input data. Unlike FanPC_CoFM, this function does not calculate error metrics against true parameters, making it suitable for simple estimation tasks.

Usage

```
FanPC_basic(data, m)
```

Arguments

data	A matrix or data frame of input data (n x p).
m	Integer. The number of principal components (factors) to extract.

Value

A list containing:

AF	Estimated factor loadings matrix (p x m).
DF	Estimated uniquenesses vector (p).
SigmahatF	The correlation matrix of the input data.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# 1. Generate synthetic data using CoFM (toy example)
sim <- CoFM(n = 200, p = 6, m = 2, type = "Clayton", param = 2.0)
obs_data <- sim$data

# 2. Apply FanPC method (extract 2 factors)
fit <- FanPC_basic(data = obs_data, m = 2)

# 3. Inspect estimates
head(fit$AF)  # Estimated loadings
fit$DF        # Estimated uniquenesses
```

FanPC_CoFM *Perform Factor Analysis via Principal Component (FanPC) for CoFM*

Description

This function estimates factor loadings and uniquenesses using a principal-component (FanPC) approach. It then compares these estimates with the true parameters (A and D) to calculate Mean Squared Errors (MSE) and relative loss metrics. This is designed to work with data generated by the CoFM function.

Usage

```
FanPC_CoFM(data, m, A, D)
```

Arguments

data	A matrix or data frame of input data (n x p). Usually the $data output from CoFM.
m	Integer. The number of principal components (factors) to extract.
A	Matrix. The true factor loadings matrix (p x m). Usually $True_Params$A from CoFM.
D	Matrix. The true uniquenesses matrix (p x p). Usually $True_Params$D from CoFM.

Value

A list containing:

AF	Estimated factor loadings matrix (p x m).
DF	Estimated uniquenesses matrix (p x p).
MSESigmaA	Mean Squared Error for factor loadings.
MSESigmaD	Mean Squared Error for uniquenesses.
LSigmaA	Relative loss metric for factor loadings.
LSigmaD	Relative loss metric for uniquenesses.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# 1. Generate toy data using CoFM
sim_result <- CoFM(n = 200, p = 6, m = 2, type = "Clayton", param = 2.0)

# 2. Extract true parameters and observed data
true_A <- sim_result$True_Params$A
true_D <- sim_result$True_Params$D
obs_data <- sim_result$data

# 3. Apply FanPC and compute error metrics
fanpc_result <- FanPC_CoFM(data = obs_data, m = 2, A = true_A, D = true_D)

# 4. Inspect results
fanpc_result$MSESigmaA
fanpc_result$MSESigmaD
head(fanpc_result$AF)
```

PC_CoFM *Perform PCA-based Factor Estimation for CoFM*

Description

This function performs Principal Component Analysis (PCA) on the correlation matrix of the data to estimate factor loadings and uniquenesses. It is designed to work with data generated by the CoFM function and calculates error metrics (MSE and relative loss) by comparing estimates against the true parameters.

Usage

```
PC_CoFM(data, m, A, D)
```

Arguments

data	A matrix or data frame of input data (n x p). Usually the $data output from CoFM.
m	Integer. The number of principal components (factors) to retain.
A	Matrix. The true factor loadings matrix (p x m). Usually $True_Params$A from CoFM.
D	Matrix. The true uniquenesses matrix (p x p). Usually $True_Params$D from CoFM.

Value

A list containing:

A2	Estimated factor loadings matrix.
D2	Estimated uniquenesses matrix.
MSESigmaA	Mean Squared Error for factor loadings.
MSESigmaD	Mean Squared Error for uniquenesses.
LSigmaA	Relative loss metric for factor loadings.
LSigmaD	Relative loss metric for uniquenesses.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# 1. Generate toy data using CoFM
sim_result <- CoFM(n = 200, p = 6, m = 2, type = "Clayton", param = 2.0)

# 2. Extract true parameters and observed data
true_A <- sim_result$True_Params$A
true_D <- sim_result$True_Params$D
obs_data <- sim_result$data

# 3. Apply PC method to estimate parameters and compute errors
pc_result <- PC_CoFM(data = obs_data, m = 2, A = true_A, D = true_D)

# 4. Inspect results
pc_result$MSESigmaA
pc_result$MSESigmaD
head(pc_result$A2)
```

poet *POET: Principal Orthogonal complEment Thresholding*

Description

Implements the POET method for large covariance matrix estimation (Fan, Liao & Mincheva, 2013). The method assumes a factor model structure, estimates the low-rank component via PCA, and applies thresholding to the sparse residual covariance matrix.

Usage

```
poet(
  X,
  r = NULL,
  r.max = 10,
  thresh = "hard",
  lambda = NULL,
  gamma = 3.7,
  delta = 1e-04,
  method.r = "IC1"
)
```

Arguments

X	Numeric matrix (T x N). T is the number of time periods (rows), N is the number of variables (columns).
r	Integer or NULL. User-specified number of factors. If NULL, r is estimated automatically using method.r.
r.max	Integer. Upper bound for the number of factors when estimating r. Default is 10.
thresh	Character. Thresholding type for the residual covariance. Options: "hard", "soft", "scad", "adapt". Default is "hard".
lambda	Numeric or NULL. Thresholding parameter. If NULL, it defaults to sqrt(log(N)/T).
gamma	Numeric. Parameter for SCAD thresholding. Default is 3.7.
delta	Numeric. Minimum eigenvalue bump to ensure positive definiteness of the residual covariance. Default is 1e-4.
method.r	Character. Method to select the number of factors if r is NULL. Options: "IC1" (Bai & Ng, 2002) or "ER" (Eigenvalue Ratio, Ahn & Horenstein, 2013).

Value

A list containing:

Sigma.poet	The estimated N x N POET covariance matrix.
Sigma.fact	The estimated N x N low-rank (factor) covariance matrix.
Sigma.resid	The estimated N x N thresholded residual covariance matrix.

F.hat	Estimated factors (T x r).
Lambda.hat	Estimated factor loadings (N x r).
r.hat	The number of factors used (estimated or specified).
R.hat	Same as Sigma.resid (for compatibility).

References

Fan, J., Liao, Y., & Mincheva, M. (2013). Large covariance estimation by thresholding principal orthogonal complements. *Journal of the Royal Statistical Society: Series B*, 75(4), 603-680.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(2025)
T_obs <- 40; N_var <- 15; r_true <- 2

# Generate a simple factor model: X = F * Lambda' + U
Lambda <- matrix(stats::rnorm(N_var * r_true), N_var, r_true)
F_scores <- matrix(stats::rnorm(T_obs * r_true), T_obs, r_true)
U <- matrix(stats::rnorm(T_obs * N_var), T_obs, N_var)
X_sim <- F_scores %*% t(Lambda) + U  # T x N

# Apply POET (choose r via IC1; use soft thresholding)
res <- poet(X_sim, r = NULL, method.r = "IC1", thresh = "soft")
res$r.hat
res$Sigma.poet[1:5, 1:5]
```

PPC_basic *Perform Basic Projected PCA (PPC) Estimation*

Description

This function performs Projected Principal Component Analysis (PPC) to estimate factor loadings and specific variances. It projects the data onto a specific subspace before performing eigen decomposition. Unlike PPC_CoFM, this function does not calculate error metrics against true parameters.

Usage

```
PPC_basic(data, m)
```

Arguments

data	A matrix or data frame of input data (n x p).
m	Integer. The number of principal components (factors) to extract.

Value

A list containing:

Apro	Estimated projected factor loadings matrix (p x m).
Dpro	Estimated projected uniquenesses vector (p).
Sigmahatpro	The covariance matrix of the projected data.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# 1. Generate toy data using CoFM
sim <- CoFM(n = 200, p = 6, m = 2, type = "Clayton", param = 2.0)
obs_data <- sim$data

# 2. Apply PPC method (extract 2 factors)
fit <- PPC_basic(data = obs_data, m = 2)

# 3. Inspect estimates
head(fit$Apro)
fit$Dpro
```

PPC_CoFM *Perform Projected PCA (PPC) Estimation for CoFM*

Description

This function performs Projected Principal Component Analysis (PPC) on the input data to estimate factor loadings and uniquenesses. It is designed to work with data generated by the CoFM function and calculates error metrics (MSE and relative loss) by comparing estimates against true parameters. The method projects data onto a subspace (using a projection operator) before performing PCA.

Usage

```
PPC_CoFM(data, m, A, D)
```

Arguments

data	A matrix or data frame of input data (n x p). Usually the $data output from CoFM.
m	Integer. The number of principal components (factors) to retain.
A	Matrix. The true factor loadings matrix (p x m). Usually $True_Params$A from CoFM.
D	Matrix. The true uniquenesses matrix (p x p). Usually $True_Params$D from CoFM.

Value

A list containing:

Ap2	Estimated factor loadings matrix (Projected).
Dp2	Estimated uniquenesses matrix (Projected).
MSESigmaA	Mean Squared Error for factor loadings.
MSESigmaD	Mean Squared Error for uniquenesses.
LSigmaA	Relative loss metric for factor loadings.
LSigmaD	Relative loss metric for uniquenesses.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)

# 1. Generate toy data using CoFM
sim_result <- CoFM(n = 200, p = 6, m = 2, type = "Clayton", param = 2.0)

# 2. Extract true parameters and observed data
true_A <- sim_result$True_Params$A
true_D <- sim_result$True_Params$D
obs_data <- sim_result$data

# 3. Apply PPC method and compute errors
ppc_result <- PPC_CoFM(data = obs_data, m = 2, A = true_A, D = true_D)

# 4. Inspect results
ppc_result$MSESigmaA
ppc_result$MSESigmaD
head(ppc_result$Ap2)
```

PPC_new *Center-then-PCA: Projection on the Orthogonal Complement of the Mean Vector*

Description

This function performs a specific type of Projected PCA where the data is projected onto the orthogonal complement of the mean vector. It effectively applies the centering projection $P = I - (1/n)J$ (where J is the all-ones matrix), optionally rescales the columns, and then performs PCA on the covariance matrix. This allows estimation of factor loadings and residual variances after removing the mean structure.

Usage

```
PPC_new(data, m)
```

Arguments

data	A matrix or data frame of input data (n x p).
m	Integer. The number of principal components (factors) to keep.

Value

A list containing:

Apro	Estimated factor loading matrix (p x m).
Dpro	Estimated residual variances (p x p diagonal matrix).
Sigmahatpro	Covariance matrix of the projected data.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(1)
dat <- matrix(stats::rnorm(200), ncol = 4)
ans <- PPC_new(data = dat, m = 2)
str(ans)
head(ans$Apro)
```

PPC_u *Projection-on-Complement PCA (Generalized)*

Description

Projects the data onto the orthogonal complement of a given vector u, eliminating the effect of u, and then performs PCA on the projected data. This is useful for removing specific trends (e.g., time trends, common market factors) before analysis.

Usage

```
PPC_u(data, m, u)
```

Arguments

data	A matrix or data frame of input data (n x p).
m	Integer. Number of principal components to retain.
u	Numeric vector of length n. The projection direction to be removed from the data. Will be normalized internally.

Value

A list containing:

Apro	Estimated factor loading matrix (p x m).
Dpro	Estimated residual variances (p x p diagonal matrix).
Sigmahatpro	Covariance matrix of the projected data.
u	The normalized projection vector used.

Examples

```
# Examples should be fast and reproducible for CRAN checks
set.seed(123)
dat <- matrix(stats::rnorm(200), ncol = 4)
u0 <- seq_len(nrow(dat))     # e.g., a linear trend to remove
res <- PPC_u(data = dat, m = 2, u = u0)
res$u
head(res$Apro)
```

Index