Package ‘tcv’

September 23, 2025

Type Package

Title Determining the Number of Factors in Poisson Factor Models via Thinning Cross-Validation

Version 0.1.0

Date 2025-09-01

Description Implements methods for selecting the number of factors in Poisson factor models, with a primary focus on Thinning Cross-Validation (TCV). The TCV method is based on the 'data thinning' technique, which probabilistically partitions each count observation into training and test sets while preserving the underlying factor structure. The Poisson factor model is then fit on the training set, and model selection is performed by comparing predictive performance on the test set. This toolkit is designed for researchers working with high-dimensional count data in fields such as genomics, text mining, and social sciences. The data thinning methodology is detailed in Dharamshi et al. (2025) <doi:10.1080/01621459.2024.2353948> and Wang et al. (2025) <doi:10.1080/01621459.2025.2546577>.

License GPL (>= 3)

Encoding UTF-8

Imports stats, GFM, countsplit, irlba

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

SystemRequirements C++17

RoxygenNote 7.3.2

URL `https://github.com/Wangzhijingwzj/tcv`

BugReports `https://github.com/Wangzhijingwzj/tcv/issues`

NeedsCompilation yes

Author Zhijing Wang [aut, cre],
Heng Peng [aut],
Peirong Xu [aut]

Maintainer Zhijing Wang `<wangzhijing@sjtu.edu.cn>`

Repository CRAN

Date/Publication 2025-09-23 07:40:02 UTC

Contents

`add_identifiability` *Enforce Identifiability Constraints on Factor Model Components*

Description

Post-processes the factor scores (H), loadings (B), and intercept (mu) to ensure a unique solution by applying SVD-based rotation. This typically enforces orthogonality constraints.

Usage

```
add_identifiability(H, B, mu)
```

Arguments

`H`	A numeric matrix of factor scores (n x q).
`B`	A numeric matrix of factor loadings (p x q).
`mu`	A numeric vector for the intercept/mean term.

Value

A list containing the transformed H, B, and mu that satisfy identifiability constraints.

`chooseFacNumber_ratio` *Estimating the Number of Factor by Eigenvalue Ratio of Natural Parameter Matrix in Generalized Factor Model.*

Description

Estimating the Number of Factor by Eigenvalue Ratio of Natural Parameter Matrix in Generalized Factor Model.

Usage

```
chooseFacNumber_ratio(
  XList,
  types,
  q_set = 1:5,
  select_method = c("SVR", "IC"),
  offset = FALSE,
  dc_eps = 1e-04,
  maxIter = 30,
  verbose = FALSE,
  parallelList = NULL
)
```

Arguments

`XList`	A list that containing an n by p matrix, where n is the number of samples, p is the number of features.
`types`	The type of data. In Poisson factor models, the type is "poisson".
`q_set`	The maximum number of factors for conducting ratio methods. Default as 5.
`select_method`	The methods to conduct GFM. Default as AM.
`offset`	Default as FALSE.
`dc_eps`	The tolerance for convergence. Default as 1e-4.
`maxIter`	The maximum iteration times. Defualt as 30.
`verbose`	Default as FALSE
`parallelList`	Whether to use parallel. Default as NULL.

Value

The number of factors estimated by ratio methods.

multiDT *Perform Thinning Cross-Validation to Select Factor Number*

Description

This function implements a K-fold cross-validation scheme based on data thinning (count splitting) to determine the optimal number of factors for a Poisson matrix factorization model.

Usage

```
multiDT(x, K = 5, rmax = 8)
```

Arguments

x A numeric matrix of count data (n x p).

K An integer, the number of folds for cross-validation. Default is 5.

rmax An integer, the maximum number of factors to test. Default is 8.

Value

A list containing two elements: - TCV: A numeric vector of total cross-validation error for each number of factors. - TICV: A numeric vector of the natural logarithm of TCV.

Examples

```
# 1. Set parameters for data generation
# Use smaller dimensions for a quick example
n <- 50 # Number of samples
p <- 30 # Number of features
true_q <- 2  # True number of factors

# 2. Generate data from a Poisson factor model
set.seed(123) # For reproducibility

# Factor matrix (scores)
FF <- matrix(rnorm(n * true_q), nrow = n, ncol = true_q)

# Loading matrix
BB <- matrix(runif(p * true_q, min = -1, max = 1), nrow = p, ncol = true_q)

# Intercept term
a <- runif(p, min = 0, max = 1)

# Enforce identifiability for a unique generating model
FF0 <- add_identifiability(FF, BB, a)$H
BB0 <- add_identifiability(FF, BB, a)$B
alpha <- add_identifiability(FF, BB, a)$mu

# Calculate the mean matrix (lambda) with some noise
lambda <- exp(FF0 %*% t(BB0) + rep(1, n) %*% t(alpha) + matrix(rnorm(n*p, 0, 0.5), n, p))

# Generate the final count data matrix 'x'
x <- matrix(rpois(n * p, lambda = as.vector(lambda)), nrow = n, ncol = p)

# 3. Run multiDT to find the best number of factors
# Use small K and rmax for a quick example run
cv_results <- multiDT(x, K = 2, rmax = 4)

# 4. Print results and select the best 'r' based on the minimum TCV
print(cv_results$TCV)
best_r <- which.min(cv_results$TCV)
```

Index