Package 'TCVS'

August 28, 2024

Type Package

Title Tree-guided compositional variable selection analysis of microbiome data

Version 1.0

Date 2024-8-27

Depends R (>= 4.3.0)

Imports CVXR,
knockoff,
GUniFrac,
cluster,
dirmult

Suggests knitr

VignetteBuilder knitr

Author Yicong Mao [aut, cre],
Zhiwen Jiang [aut],
Tianying Wang[aut],
Yi-Juan Hu[aut],
Xiang Zhan[aut]

Maintainer Yicong Mao <ycmao@hsc.pku.edu.cn>

Description The TCVS package implements the Tree-guided Compositional Variable Selection method to identify outcome-associated components within high-dimensional microbial compositional data. This method enhances variable selection in microbiome data by incorporating auxiliary knockoff copies of microbiome features, treated as noise, and leveraging tree structure. It integrates hierarchical taxonomic tree information inherent to microbial taxa, refining the selection process. Furthermore, TCVS employs auxiliary knockoff copies of microbiome features to fine-tune the algorithm, achieving more precise outcomes. This approach is crucial for improving selection accuracy and proves advantageous, especially when the signal strength exceeds minimal thresholds.

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.2

LazyData true

R topics documented:

`BIC_TCVS` *BIC Calculation for CVXR Optimization*

Description

This function calculates the Bayesian Information Criterion (BIC) for a given set of parameters using the CVXR package.

Usage

```
BIC_TCVS(
  Z,
  Y,
  p,
  group,
  maxlam,
  minlam,
  nlam,
  lambda_seq,
  A0,
  CoefNormalization = T
)
```

Arguments

`Z`	the design matrix matrix.
`Y`	the response vector.
`p`	the number of features the for the simulated data.
`group`	the taxonomic structure of OTUs.
`maxlam`	the maximum lambda value for the tuning parameter lambda candidates.
`minlam`	the minimum lambda value for the tuning parameter lambda candidates.
`nlam`	the number of lambda value candidates.
`lambda_seq`	the sequence of lambda values. If NULL, a sequence will be generated.
`A0`	the constraint matrix.
`CoefNormalization`	Logical. Whether to normalize the coefficients within each group. The default is T.

Value

A list containing the following elements:

- total.results - A list of results for each lambda value.
- best_lambda_BIC - The lambda value with the minimum BIC.

`cvxr.result` *CVXR Optimization Result*

Description

This function performs an optimization using the CVXR package to solve an augmented regression problem with the penalty.

Usage

```
cvxr.result(y, Z, A0, lambda, p, group, CoefNormalization = T)
```

Arguments

`y`	the response vector.
`Z`	the design matrix matrix.
`A0`	The constraint matrix, which enforces the condition that the coefficients of the original features and their knockoff counterparts sum to zero, respectively.
`lambda`	the regularization parameter.
`p`	the number of features the for the simulated compositional data.
`group`	the taxonomic structure of OTUs.
`CoefNormalization`	Logical. Whether to normalize the coefficients within each group. The default is T.

Value

The estimated coefficients from the optimization.

`dd` *Example Microbiome Dataset*

Description

A dataset containing the counts of 856 taxa from an upper-respiratory-tract microbiome study.

Usage

```
data(DirMultOutput)
```

Format

An object of class ‘list‘.

get.all.knockoff *Generate Knockoff Copy Matrix*

Description

This function generates the knockoff copies using a function from knockoff.

Usage

```
get.all.knockoff(Z, seed = 2023)
```

Arguments

Z	the original matrix.
seed	the seed for random number generation. The default is 2023.

Value

The knockoff copy matrix.

get.all.OTU *Generate OTU Matrix*

Description

This function generates the OTU matrix.

Usage

```
get.all.OTU(
  n,
  p,
  trans = c(0, 1, 2),
  type,
  normalizeMethod,
  pseudocount,
  seed = 2023
)
```

Arguments

n	the number of samples for the simulated compositional data.
p	the number of features the for the simulated compositional data.
trans	an integer specifying the type of transformation applied to the data. The options are 0 (no transformation), 1 (log transformation), and 2 (clr transformation).
type	the type of the simulated data to be generated. The options are "Lognormal_previous" and "Dirmult".
normalizeMethod	the normalization method.

pseudocount	the pseudocount value.
seed	an integer value used to set the seed of the random number generator for ensuring reproducibility. The default is 2023.

Value

The generated OTU matrix.

Examples

```
# Example usage
library(GUniFrac)
n = 200
p = 60
loop_start_index = 1
seed = 2023
type = "Dirmult" # type to generate compositional data
normalizeMethod = "Rowsum"
pseudocount = 0.5
X <- get.all.OTU(n, p, trans = 0, type, normalizeMethod, pseudocount, seed = seed)
```

---

get.all.y *Generate Response Variable Based on the CLR matrix Z*

Description

This function generates a response variable 'y' as a linear combination of predictors specified in matrix 'Z', influenced by a vector of coefficients 'beta'. The coefficients are determined based on predefined settings and optionally modified by a slack factor. Random noise is added to the response to simulate more realistic scenarios.

Usage

```
get.all.y(Z, setting = c(1, 2), beta.slack.factor = 1, seed = 2023)
```

Arguments

Z	the CLR transforamtion matrix (design matrix).
setting	a numeric vector indicating the setting for coefficient generation. Setting 1 generates a fixed set of coefficients with specific non-zero values. Setting 2 generates coefficients by sampling from a uniform distribution.
beta.slack.factor	an optional numeric factor to adjust the coefficients by a given factor. If not provided, coefficients are used as defined by the 'setting'. The default is 1.
seed	an integer value to set the seed for random number generation to ensure reproducibility.

Value

A numeric vector representing the generated response variable 'y'.

Examples

```
Z <- matrix(rnorm(100 * 20), ncol = 20)
y <- get.all.y(Z, setting = 1, beta.slack.factor = 1, seed = 2023)
```

`Group.lasso` *Group Lasso Penalty with L2 Norm (Without Square)*

Description

This function computes the penalty term in the augmented problem incorporating auxiliary knockoff copies and tree structure with L2 norm (without square) for a given set of coefficients, groups, and a regularization parameter.

Usage

```
Group.lasso(beta, group, lambda, p, CoefNormalization = T)
```

Arguments

`beta`	a numeric vector of coefficients.
`group`	the taxonomic structure of OTUs.
`lambda`	the regularization parameter.
`p`	the number of features the for the simulated compositional data.
`CoefNormalization`	Logical. Whether to normalize the coefficients within each group. The default is T.

Value

The computed penalty.

`P` *Taxonomy Structure of OTUs in our example*

Description

The taxonomy structure of OTUs. The first 2p rows of the matrix P correspond to the l_1 penalty applied to the CLR (Centered Log-Ratio) covariates followed by their knockoff copies. The remaining rows correspond to the tree structure associated with the CLR covariates and their knockoff copies, following the same order of covariates first and knockoff copies second.

Usage

```
data(P_60)
```

Format

An object of class ‘matrix‘.

TCVS *A Tree-guided Compositional Variable Selection method (TCVS)*

Description

This function allows you to identify outcome-associated components (OTUs) within high-dimensional microbial compositional data with hierarchical taxonomic tree information inherent among microbial taxa.

Usage

```
TCVS(X, Z, y, P, method = "BIC", maxlam, minlam, nlam, q = 0.05, seed = 2023)
```

Arguments

X	the compositional covariate matrix.
Z	the CLR transforamtion matrix (design matrix).
y	the response vector.
P	the taxonomic structure of OTUs.
method	the method used for selecting the tuning parameter. The default is "BIC".
maxlam	the maximum lambda value for the tuning parameter lambda candidates.
minlam	the minimum lambda value for the tuning parameter lambda candidates.
nlam	the number of lambda value candidates.
q	A threshold value between 0 and 1. The default is 0.05.
seed	an integer value used to set the seed of the random number generator for ensuring reproducibility. The default is 2023.

Value

A list containing the following elements:

- TCVS.beta.hat - The estimated coefficients from the TCVS method.
- S.TCVS - The selected variables from the TCVS method.

Examples

```
# Example usage
library(GUniFrac)
n = 200
p = 60
sim.setting = 1
method = "BIC"
seed = 2023
type = "Dirmult" # type to generate compositional data
normalizeMethod = "Rowsum"
beta.slack.factor = NULL
pseudocount = 0.5
maxlam = 0.1
minlam = 1e-7
nlam = 2
```

```
q = 0.05
data(P_60, package = "TCVS") # P matrix
X <- get.all.OTU(n, p, trans = 0, type, normalizeMethod, pseudocount, seed = seed)
# clr transformation
Z <- get.all.OTU(n, p, trans = 2, type, normalizeMethod, pseudocount, seed = seed)
y <- get.all.y(Z, setting = sim.setting, beta.slack.factor, seed = seed)
result.TCVS <- TCVS(
  X = X,
  Z = Z,
  y = y,
  P = P,
  method = method,
  maxlam = maxlam,
  minlam = minlam,
  nlam = nlam,
  q = q,
  seed = seed
)
```

`TCVS_Generate_X_Z` *Generate compositional matrix X and CLR transformation matrix Z*

Description

This function generates the X and Z matrices based on the specified type and parameters.

Usage

```
TCVS_Generate_X_Z(
  n,
  p,
  type = c("Lognormal_previous", "Dirmult"),
  normalizeMethod = "Rowsum",
  pseudocount
)
```

Arguments

`n`	the number of samples for the simulated compositional data.
`p`	the number of features the for the simulated compositional data.
`type`	the type of the simulated data to be generated. The options are "Lognormal_previous" and "Dirmult".
`normalizeMethod`	the method of normalization when type = "Dirmult". The default is "Rowsum".
`pseudocount`	a pseudo-count value added to the count matrix to prevent zero values in the read count.

Value

A list containing the following matrices:

- X - The compositional matrix.
- log_X - The logarithm of the compositional matrix.
- Z - The CLR transformation matrix.

Examples

```
# Example usage for Dirmult type
result <- TCVS_Generate_X_Z(n = 200, p = 60, type="Lognormal_previous",
normalizeMethod, pseudocount = 0.5)
```

Index