Package ‘argminCS’

July 14, 2025

Type Package

Title Argmin Inference over a Discrete Candidate Set

Version 1.1.0

Date 2025-07-07

Description Provides methods to construct frequentist confidence sets with valid marginal coverage for identifying the population-level argmin or argmax based on IID data. For instance, given an n by p loss matrix—where n is the sample size and p is the number of models—the CS.argmin() method produces a discrete confidence set that contains the model with the minimal (best) expected risk with desired probability. The argmin.HT() method helps check if a specific model should be included in such a confidence set. The main implemented method is proposed by Tianyu Zhang, Hao Lee and Jing Lei (2024) ``Winners with confidence: Discrete argmin inference with an application to model selection''.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

RdMacros Rdpack

Imports BSDA, glue, LDATS, MASS, methods, Rdpack, stats, withr

URL https://github.com/xu3cl4/argminCS

NeedsCompilation no

Author Tianyu Zhang [aut],
Hao Lee [aut, cre, cph],
Jing Lei [aut]

Maintainer Hao Lee <haolee@andrew.cmu.edu>

Repository CRAN

Date/Publication 2025-07-14 16:30:09 UTC

Contents

`argmax.HT` *A wrapper to perform argmax hypothesis test.*

Description

This function performs a hypothesis test to evaluate whether a given dimension may be the argmax. It internally negates the data and reuses the implementation from `argmin.HT`.

Usage

```
argmax.HT(data, r = NULL, method = "softmin.LOO", ...)
```

Arguments

`data`	(1) A n by p matrix of raw samples (for GTA), or (2) A n by (p-1) difference matrix (for SML, HML, NS, MT). Each row is a sample.
`r`	The dimension of interest for testing; defaults to NULL. Required for GTA.
`method`	A string indicating the method to use. Defaults to 'softmin.LOO'. See **Details** for supported methods and abbreviations.
`...`	Additional arguments passed to `argmin.HT.LOO`, `argmin.HT.MT`, `argmin.HT.nonsplit`, or `argmin.HT.gupta`.

Details

The supported methods include:

`softmin.LOO (SML)`	Leave-one-out algorithm using exponential weighting.
`argmin.LOO (HML)`	A variant of SML that uses hard argmin instead of exponential weighting. Not recommended.
`nonsplit (NS)`	Variant of SML without data splitting. Requires a fixed lambda value. Not recommended.
`Bonferroni (MT)`	Multiple testing using Bonferroni correction.

Gupta (GTA) The method from Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Tech*

Value

A character string: 'Accept' or 'Reject', indicating whether the dimension could be an argmax, and relevant statistics.

References

Chernozhukov V, Chetverikov D, Kato K (2013). "Testing many moment inequalities." RePEc. IDEAS Working Paper Series.

Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Technometrics*, **7**(2), 225–245. doi:10.1080/00401706.1965.10490251.

Futschik A, Pflug G (1995). "Confidence Sets for Discrete Stochastic Optimization." *Annals of Operations Research*, **56**(1), 95–108. doi:10.1007/BF02031702.

Examples

```
set.seed(108)
n <- 200
p <- 20
mu <- (1:p)/p
cov <- diag(p)
data <- MASS::mvrnorm(n, mu, cov)

## Define the dimension of interest
r <- 4

## Construct difference matrix for dimension r
difference.matrix.r <- matrix(rep(data[, r], p - 1), ncol = p - 1, byrow = FALSE) - data[, -r]

## softmin.LOO (SML)
argmax.HT(difference.matrix.r)

## use seed
argmax.HT(difference.matrix.r, seed=19)

## With known true difference
true.mean.diff <- mu[r] - mu[-r]
argmax.HT(difference.matrix.r, true.mean = true.mean.diff)

## Without scaling
argmax.HT(difference.matrix.r, scale.input = FALSE)

## With a user-specified lambda
argmax.HT(difference.matrix.r, lambda = sqrt(n) / 2.5)

## Add a seed for reproducibility
argmax.HT(difference.matrix.r, seed = 17)
```

```
## argmin.LOO (HML)
argmax.HT(difference.matrix.r, method = "HML")

## nonsplit method
argmax.HT(difference.matrix.r, method = "NS", lambda = sqrt(n)/2.5)

## Bonferroni method (choose t test for normal data)
argmax.HT(difference.matrix.r, method = "MT", test = "t")

## Gupta method (pass full data matrix)
critical.val <- get.quantile.gupta.selection(p = length(mu))
argmax.HT(data, r, method = "GTA", critical.val = critical.val)
```

`argmin.HT` *A wrapper to perform argmin hypothesis test.*

Description

This is a wrapper to perform hypothesis test to see if a given dimension may be an argmin. Multiple methods are supported.

Usage

```
argmin.HT(data, r = NULL, method = "softmin.LOO", ...)
```

Arguments

`data`	(1) A n by p data matrix for (GTA); each of its row is a p-dimensional sample, or (2) A n by (p-1) difference matrix for (SML, HML, NS, MT); each of its row is a (p-1)-dimensional sample differences
`r`	The dimension of interest for hypothesis test; defaults to NULL. (Only needed for GTA)
`method`	A string indicating the method for hypothesis test; defaults to 'softmin.LOO'. Passing an abbreviation is allowed. For the list of supported methods and their abbreviations, see Details.
`...`	Additional arguments to argmin.HT.LOO, lambda.adaptive.enlarge, is.lambda.feasible.LOO, argmin.HT.MT, argmin.HT.gupta. A correct argument name needs to be specified if it is used.

Details

The supported methods include:

`softmin.LOO (SML)` LOO (leave-one-out) algorithm, using the exponential weightings. Proposed by Zhang T, Lee H, Lei J (

`argmin.LOO (HML)`	A variant of SML, but it uses (hard) argmin rather than exponential weighting. The method is not recon
`nonsplit (NS)`	A variant of SML, but no splitting is involved. One needs to pass a fixed lambda value as a required add
`Bonferroni (MT)`	Multiple testing with Bonferroni's correction.
`Gupta (GTA)`	The method in Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Techno*

Value

'Accept' or 'Reject'. A string indicating whether the given dimension could be an argmin (Accept) or not (Reject), and relevant statistics.

References

Zhang T, Lee H, Lei J (2024). "Winners with confidence: Discrete argmin inference with an application to model selection." *arXiv preprint arXiv:2408.02060*.

Chernozhukov V, Chetverikov D, Kato K (2013). "Testing many moment inequalities." RePEc. IDEAS Working Paper Series.

Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Technometrics*, **7**(2), 225–245. doi:10.1080/00401706.1965.10490251.

Futschik A, Pflug G (1995). "Confidence Sets for Discrete Stochastic Optimization." *Annals of Operations Research*, **56**(1), 95–108. doi:10.1007/BF02031702.

Examples

```
r <- 4
n <- 200
p <- 20
mu <- (1:p)/p
cov <- diag(length(mu))
set.seed(108)
data <- MASS::mvrnorm(n, mu, cov)
sample.mean <- colMeans(data)

## softmin.LOO
difference.matrix.r <- matrix(rep(data[,r], p-1), ncol=p-1, byrow=FALSE) - data[,-r]
argmin.HT(difference.matrix.r)

## use seed
argmin.HT(difference.matrix.r, seed=19)

# provide centered test statistic (to simulate asymptotic normality)
true.mean.difference.r <- mu[r] - mu[-r]
argmin.HT(difference.matrix.r, true.mean=true.mean.difference.r)

# keep the data unstandardized
argmin.HT(difference.matrix.r, scale.input=FALSE)
```

```
# use an user-specified lambda
argmin.HT(difference.matrix.r, lambda=sqrt(n)/2.5)

# add a seed
argmin.HT(difference.matrix.r, seed=19)

## argmin.LOO/hard min
argmin.HT(difference.matrix.r, method='HML')

## nonsplit
argmin.HT(difference.matrix.r, method='NS', lambda=sqrt(n)/2.5)

## Bonferroni (choose t test because of normal data)
argmin.HT(difference.matrix.r, method='MT', test='t')
## z test
argmin.HT(difference.matrix.r, method='MT', test='z')

## Gupta
critical.val <- get.quantile.gupta.selection(p=length(mu))
argmin.HT(data, r, method='GTA', critical.val=critical.val)
```

`argmin.HT.gupta` *Perform argmin hypothesis test using Gupta's method.*

Description

Test whether a dimension is the argmin, using the method in (Gupta 1965).

Usage

```
argmin.HT.gupta(
  data,
  r,
  sample.mean = NULL,
  stds = NULL,
  critical.val = NULL,
  alpha = 0.05,
  ...
)
```

Arguments

`data`	A n by p data matrix; each of its row is a p-dimensional sample.
`r`	The dimension of interest for hypothesis test.
`sample.mean`	The sample mean of the n samples in data; defaults to NULL. It can be calculated via colMeans(data). If performing multiple tests across dimensions, pre-computing `sample.mean` and `critical.val` can significantly reduce computation time.

`stds`	A vector of the same (population) standard deviations for all dimensions; defaults to a vector of 1's. These are used to standardize the sample means.
`critical.val`	The quantile for the hypothesis test; defaults to NULL. It can be calculated via get.quantile.gupta.selection. If your experiment involves hypothesis testing over more than one dimension, pass a quantile to speed up computation.
`alpha`	The significance level of the hypothesis test; defaults to 0.05.
`...`	Additional argument to get.quantile.gupta.selection. A correct argument name needs to be specified if it is used.

Value

A list containing:

`test.stat`	The test statistic
`.critical.value`	The critical value for the hypothesis test. Being greater than it leads to a rejection.
`ans`	'Reject' or 'Accept'

Note

This method requires independence among the dimensions.

References

Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Technometrics*, **7**(2), 225–245. doi:10.1080/00401706.1965.10490251.

Futschik A, Pflug G (1995). "Confidence Sets for Discrete Stochastic Optimization." *Annals of Operations Research*, **56**(1), 95–108. doi:10.1007/BF02031702.

`argmin.HT.LOO` *Perform argmin hypothesis test.*

Description

Test if a dimension may be argmin, using the LOO (leave-one-out) algorithm in Zhang et al 2024.

Usage

```
argmin.HT.LOO(
  difference.matrix,
  sample.mean = NULL,
  min.algor = "softmin",
  lambda = NULL,
  const = 2.5,
```

```
  enlarge = TRUE,
  alpha = 0.05,
  true.mean.difference = NULL,
  output.weights = FALSE,
  scale.input = TRUE,
  seed = NULL,
  ...
)
```

Arguments

difference.matrix
: A n by (p-1) difference data matrix (reference dimension - the rest); each of its row is a (p-1)-dimensional vector of differences.

sample.mean
: The sample mean of differences; defaults to NULL. It can be calculated via colMeans(difference.matrix).

min.algor
: The algorithm to compute the test statistic by weighting across dimensions; 'softmin' uses exponential weighting, while 'argmin' picks the largest mean coordinate directly. Defaults to 'softmin'.

lambda
: The real-valued tuning parameter for exponential weightings (the calculation of softmin); defaults to NULL. If lambda=NULL (recommended), the function would determine a lambda value in a data-driven way.

const
: The scaling constant for initial data-driven lambda

enlarge
: A boolean value indicating if the data-driven lambda should be determined via an iterative enlarging algorithm; defaults to TRUE.

alpha
: The significance level of the hypothesis test; defaults to 0.05.

true.mean.difference
: The population mean of the differences. (Optional); used to compute a centered test statistic for simulation or diagnostic purposes.

output.weights
: A boolean variable specifying whether the exponential weights should be outputted; defaults to FALSE.

scale.input
: A boolean variable specifying whether the input difference matrix should be standardized. Defaults to TRUE

seed
: (Optional) If provided, used to seed the random sampling (for reproducibility).

...
: Additional arguments to lambda.adaptive.enlarge, is.lambda.feasible.LOO.

Value

A list containing:

test.stat.scale
: The scaled test statistic

critical.value
: The critical value for the hypothesis test. Being greater than it leads to a rejection.

std
: The standard deviation estimate.

ans	A character string: either 'Reject' or 'Accept', depending on the test outcome.
lambda	The lambda used in the hypothesis testing.
lambda.capped	Boolean variable indicating the data-driven lambda has reached the large threshold n^5
residual.slepian	The final approximate first order stability term for the data-driven lambda.
variance.bound	The final variance bound for the data-driven lambda.
test.stat.centered	(Optional) The centered test statistic, computed only if `true.mean.difference` is provided.
exponential.weights	(Optional) A (n by p-1) matrix storing the exponential weightings in the test statistic.

---

argmin.HT.MT *Perform argmin hypothesis test.*

Description

Test if a dimension may be argmin, using multiple testing with Bonferroni's correction.

Usage

```
argmin.HT.MT(difference.matrix, sample.mean = NULL, test = "z", alpha = 0.05)
```

Arguments

difference.matrix	A n by (p-1) difference data matrix (reference dimension - the rest); each of its row is a (p-1)-dimensional vector of differences.
sample.mean	The sample mean of differences; defaults to NULL. It can be calculated via colMeans(difference.matrix).
test	The test to perform: 't' or 'z'; defaults to 'z'. If the data are assumed normally distributed, use 't'; otherwise 'z'.
alpha	The significance level of the hypothesis test; defaults to 0.05.

Value

A list containing:

p.val	p value without Bonferroni's correction.
. critical.value	The critical value for the hypothesis test. Being less than it leads to a rejection.

`ans`	'Reject' or 'Accept'

`argmin.HT.nonsplit` *Perform argmin hypothesis test.*

Description

Test if a dimension may be argmin without any splitting.

Usage

```
argmin.HT.nonsplit(
  difference.matrix,
  lambda,
  sample.mean = NULL,
  alpha = 0.05,
  scale.input = TRUE
)
```

Arguments

`difference.matrix`	A n by (p-1) difference data matrix (reference dimension - the rest); each of its row is a (p-1)-dimensional vector of differences.
`lambda`	The real-valued tuning parameter for exponential weightings (the calculation of softmin).
`sample.mean`	The sample mean of differences; defaults to NULL. It can be calculated via colMeans(difference.matrix).
`alpha`	The significance level of the hypothesis test; defaults to 0.05.
`scale.input`	A boolean variable specifying whether the input difference matrix should be standardized defaults to TRUE

Details

This method is not recommended, given its poor performance when p is small.

Value

A list containing:

`test.stat.scale`	The scaled test statistic
`.critical.value`	The critical value for the hypothesis test. Being greater than it leads to a rejection.

`std`	The standard deviation estimate.
`ans`	'Reject' or 'Accept'

`CS.argmax` *Construct a discrete confidence set for argmax.*

Description

This is a wrapper to construct a confidence set for the argmax by negating the input and reusing `CS.argmin`.

Usage

```
CS.argmax(data, method = "softmin.LOO", alpha = 0.05, ...)
```

Arguments

`data`	An $n \times p$ matrix; each row is a p-dimensional sample.
`method`	A string indicating the method to use; defaults to 'softmin.LOO'. Can be abbreviated (e.g., 'SML' for 'softmin.LOO'). See Details for full list.
`alpha`	Significance level. The function returns a $1 - \alpha$ confidence set.
`...`	Additional arguments passed to corresponding testing functions.

Details

The supported methods include:

`softmin.LOO (SML)`	Leave-one-out algorithm using exponential weighting.
`argmin.LOO (HML)`	Variant of SML that uses hard argmin instead of soft weighting. Not recommended.
`nonsplit (NS)`	Variant of SML without data splitting. Requires a fixed lambda value. Not recommended.
`Bonferroni (MT)`	Multiple testing using Bonferroni correction.
`Gupta (GTA)`	The method of Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Techno*
`Futschik (FCHK)`	A two-step method from Futschik A, Pflug G (1995). "Confidence Sets for Discrete Stochastic Optimiz

Value

A vector of indices (1-based) representing the confidence set for the argmax.

References

Zhang T, Lee H, Lei J (2024). "Winners with confidence: Discrete argmin inference with an application to model selection." *arXiv preprint arXiv:2408.02060*.

Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Technometrics*, **7**(2), 225–245. doi:10.1080/00401706.1965.10490251.

Futschik A, Pflug G (1995). "Confidence Sets for Discrete Stochastic Optimization." *Annals of Operations Research*, **56**(1), 95–108. doi:10.1007/BF02031702.

Chernozhukov V, Chetverikov D, Kato K (2013). "Testing many moment inequalities." RePEc. IDEAS Working Paper Series.

Examples

```
set.seed(108)
n <- 200
p <- 20
mu <- (1:p)/p
cov <- diag(p)
data <- MASS::mvrnorm(n, mu, cov)

## softmin.LOO (SML)
CS.argmax(data)

## argmin.LOO (HML)
CS.argmax(data, method = "HML")

## nonsplit (NS) - requires lambda
CS.argmax(data, method = "NS", lambda = sqrt(n)/2.5)

## Bonferroni (MT) - t test default
CS.argmax(data, method = "MT", test = "t")

## Gupta (GTA)
CS.argmax(data, method = "GTA")

## Futschik (FCHK) with default alpha.1 and alpha.2
CS.argmax(data, method = "FCHK")

## Futschik (FCHK) with user-specified alpha.1 and alpha.2
alpha.1 <- 0.001
alpha.2 <- 1 - (0.95 / (1 - alpha.1))
CS.argmax(data, method = "FCHK", alpha.1 = alpha.1, alpha.2 = alpha.2)
```

`CS.argmin` *Construct a discrete confidence set for argmin.*

Description

This is a wrapper to construct a discrete confidence set for argmin. Multiple methods are supported.

Usage

```
CS.argmin(data, method = "softmin.LOO", alpha = 0.05, ...)
```

Arguments

`data`	A n by p data matrix; each row is a p-dimensional sample.
`method`	A string indicating the method used to construct the confidence set. Defaults to 'softmin.LOO'. Can be abbreviated (e.g., 'SML' for 'softmin.LOO'). See **Details** for available methods and abbreviations.
`alpha`	The significance level; defaults to 0.05. The function produces a $1 - \alpha$ confidence set.
`...`	Additional arguments to argmin.HT.LOO, lambda.adaptive.enlarge, is.lambda.feasible.LOO, argmin.HT.MT, argmin.HT.gupta. A correct argument name needs to be specified if it is used.

Details

The supported methods include:

`softmin.LOO (SML)`	Leave-one-out algorithm using exponential weighting. Proposed by Zhang T, Lee H, Lei J (2024). “Wi
`argmin.LOO (HML)`	A variant of SML that uses hard argmin instead of exponential weighting. Not recommended.
`nonsplit (NS)`	A variant of SML without data splitting. Requires a fixed lambda value as an additional argument. Not
`Bonferroni (MT)`	Multiple testing using Bonferroni correction.
`Gupta (GTA)`	The method proposed by Gupta (1965). Requires independence and the same population standard devia
`Futschik (FCHK)`	A two-step method from Futschik and Pflug (1995). Requires independence and the same population st

Value

A vector of indices (1-based) representing the (1 - alpha) confidence set.

References

Zhang T, Lee H, Lei J (2024). “Winners with confidence: Discrete argmin inference with an application to model selection.” *arXiv preprint arXiv:2408.02060*.

Chernozhukov V, Chetverikov D, Kato K (2013). “Testing many moment inequalities.” RePEc. IDEAS Working Paper Series.

Gupta SS (1965). “On Some Multiple Decision (Selection and Ranking) Rules.” *Technometrics*, **7**(2), 225–245. doi:10.1080/00401706.1965.10490251.

Futschik A, Pflug G (1995). “Confidence Sets for Discrete Stochastic Optimization.” *Annals of Operations Research*, **56**(1), 95–108. doi:10.1007/BF02031702.

Examples

```
r <- 4
n <- 200
mu <- (1:20)/20
cov <- diag(length(mu))
set.seed(108)
data <- MASS::mvrnorm(n, mu, cov)
sample.mean <- colMeans(data)

## softmin.LOO
CS.argmin(data)

## use seed
CS.argmin(data, seed=13)

## argmin.LOO
CS.argmin(data, method='HML')

## nonsplit
CS.argmin(data, method='NS', lambda=sqrt(n)/2.5)

## Bonferroni (choose t test because of normal data)
CS.argmin(data, method='MT', test='t')

## Gupta
CS.argmin(data, method='GTA')

## Futschik two-step method
# default alpha.1, alpha.2
CS.argmin(data, method='FCHK')

alpha.1 <- 0.0005
alpha.2 <- 1 - (0.95/(1 - alpha.1))
CS.argmin(data, method='FCHK', alpha.1=0.0005, alpha.2=alpha.2)
```

`find.sub.argmin` *Get the index of the smallest dimension apart from an index*

Description

Get the index of the smallest dimension apart from an index

Usage

```
find.sub.argmin(nums, idx, seed = NULL)
```

Arguments

nums	A vector of numbers
idx	An index to be excluded
seed	(Optional) If provided, used to seed the random sampling (for reproducibility).

Value

The index of the second smallest dimension (as an integer).

Examples

```
nums <- c(1,3,2)
find.sub.argmin(nums,1)
## return 3

nums <- c(1,1,2)
find.sub.argmin(nums,1)
## return 2
```

---

get.quantile.gupta.selection

Generate the quantile used for the selection procedure in (Gupta 1965).

Description

Generate the quantile used for the selection procedure in (Gupta 1965) by Monte Carlo estimation.

Usage

```
get.quantile.gupta.selection(p, alpha = 0.05, N = 1e+05)
```

Arguments

p	The number of dimensions in your data matrix.
alpha	The level of the upper quantile; defaults to 0.05 (95% percentile).
N	The number of Monte Carlo repetitions; defaults to 100000.

Value

A list containing:

critica.val	The 1 - alpha upper quantile.

Note

The quantile is pre-calculated for some common configurations of (p, alpha)

References

Gupta SS (1965). "On Some Multiple Decision (Selection and Ranking) Rules." *Technometrics*, **7**(2), 225–245. doi:10.1080/00401706.1965.10490251.

Futschik A, Pflug G (1995). "Confidence Sets for Discrete Stochastic Optimization." *Annals of Operations Research*, **56**(1), 95–108. doi:10.1007/BF02031702.

Examples

```
get.quantile.gupta.selection(p=10)

get.quantile.gupta.selection(p=100)
```

`is.lambda.feasible.LOO` *Check the feasibility of a tuning parameter λ for LOO algorithm.*

Description

Check the feasibility of a tuning parameter λ for LOO algorithm by examining whether its resulting $\nabla_i K_j$ is less than a threshold value, i.e., the first order stability is likely achieved. For further details, we refer to the paper Zhang et al 2024.

Usage

```
is.lambda.feasible.LOO(
  lambda,
  scaled.difference.matrix,
  sample.mean = NULL,
  threshold = 0.08,
  n.pairs = 100,
  seed = NULL
)
```

Arguments

`lambda` The real-valued tuning parameter for exponential weightings (the calculation of softmin).

`scaled.difference.matrix`
A n by (p-1) difference scaled.difference.matrix matrix after column-wise scaling (reference dimension - the rest); each of its row is a (p-1)-dimensional vector of differences.

`sample.mean` The sample mean of the n samples in scaled.difference.matrix; defaults to NULL. It can be calculated via colMeans(scaled.difference.matrix). If your experiment involves hypothesis testing over more than one dimension, pass sample.mean=colMeans(scaled.difference to speed up computation.

threshold	A threshold value to examine if the first order stability is likely achieved; defaults to 0.08. As its value gets smaller, the first order stability tends to increase while power might decrease.
n.pairs	The number of (i, j) pairs for estimation; defaults to 100.
seed	(Optional) An integer-valued seed for subsampling.

Value

A boolean value indicating if the given λ likely gives the first order stability.

---

lambda.adaptive.enlarge

Iteratively enlarge a tuning parameter λ in a data-driven way.

Description

Iteratively enlarge a tuning parameter λ to enhance the power of hypothesis testing. The iterative algorithm ends when an enlarged λ unlikely yields the first order stability.

Usage

```
lambda.adaptive.enlarge(
  lambda,
  scaled.difference.matrix,
  sample.mean = NULL,
  mult.factor = 2,
  verbose = FALSE,
  seed = NULL,
  ...
)
```

Arguments

lambda	The real-valued tuning parameter for exponential weightings (the calculation of softmin).
scaled.difference.matrix	A n by (p-1) difference scaled.difference.matrix matrix after column-wise scaling (reference dimension - the rest); each of its row is a (p-1)-dimensional vector of differences.
sample.mean	The sample mean of the n samples in scaled.difference.matrix; defaults to NULL. It can be calculated via colMeans(scaled.difference.matrix). If your experiment involves hypothesis testing over more than one dimension, pass sample.mean=colMeans(scaled.difference to speed up computation.
mult.factor	In each iteration, λ would be multiplied by mult.factor to yield an enlarged λ; defaults to 2.

`verbose`	A boolean value indicating if the number of iterations should be printed to console; defaults to FALSE.
`seed`	(Optional) If provided, used to seed for tie-breaking (for reproducibility).
`...`	Additional arguments to is.lambda.feasible.LOO.

Value

A list containing:

`lambda`	The final (enlarged) lambda that is still feasible.
`capped`	Logical, `TRUE` if the enlargement was capped due to reaching the threshold.
`residual.slepian`	Residual value from the feasibility check at the final lambda.
`variance.bound`	Variance bound used in the final feasibility check.

Examples

```
# Simulate data
set.seed(123)
r <- 4
n <- 200
mu <- (1:20)/20
cov <- diag(length(mu))
set.seed(108)
data <- MASS::mvrnorm(n, mu, cov)
sample.mean <- colMeans(data)
diff.mat <- get.difference.matrix(data, r)
sample.mean.r <- get.sample.mean.r(sample.mean, r)
lambda <- lambda.adaptive.LOO(diff.mat, sample.mean=sample.mean.r)

# Run the enlargement algorithm
res <- lambda.adaptive.enlarge(lambda, diff.mat, sample.mean=sample.mean.r)
res
# with a seed
res <- lambda.adaptive.enlarge(lambda, diff.mat, sample.mean=sample.mean.r, seed=3)
res
```

`lambda.adaptive.LOO` *Generate a scaled.difference.matrix-driven λ for LOO algorithm.*

Description

Generate a scaled.difference.matrix-driven λ for LOO algorithm motivated by the derivation of the first order stability. For its precise definition, we refer to the paper Zhang et al 2024.

Usage

```
lambda.adaptive.LOO(
  scaled.difference.matrix,
  sample.mean = NULL,
  const = 2.5,
  seed = NULL
)
```

Arguments

`scaled.difference.matrix`
: A n by (p-1) difference scaled.difference.matrix matrix after column-wise scaling (reference dimension - the rest); each of its row is a (p-1)-dimensional vector of differences.

`sample.mean`
: The sample mean of the n samples in scaled.difference.matrix; defaults to NULL. It can be calculated via colMeans(scaled.difference.matrix).

`const`
: A scaling constant for the scaled.difference.matrix driven λ; defaults to 2.5. As its value gets larger, the first order stability tends to increase while power might decrease.

`seed`
: (Optional) If provided, used to seed for tie-breaking (for reproducibility).

Value

A scaled.difference.matrix-driven λ for LOO algorithm.

Examples

```
# Simulate data
set.seed(123)
r <- 4
n <- 200
mu <- (1:20)/20
cov <- diag(length(mu))
set.seed(108)
data <- MASS::mvrnorm(n, mu, cov)
sample.mean <- colMeans(data)
diff.mat <- get.difference.matrix(data, r)
sample.mean.r <- get.sample.mean.r(sample.mean, r)
lambda <- lambda.adaptive.LOO(diff.mat, sample.mean=sample.mean.r)
```

Index