Package ‘rgTest’
August 14, 2023

Title Robust Graph-Based Two-Sample Test
Version 0.1
Description Useful tools for determining whether two samples are from the same distribution. Utilizes a robust method to address the problematic structure of the similarity graph constructed from high-dimensional data. The method is provided in Yichuan Bai and Lynna Chu (2023) <arXiv:2307.12325>.
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.2.3
Imports ade4, stats
Suggests knitr, rmarkdown, testthat (>= 3.0.0)
Config/testthat/edition 3
VignetteBuilder knitr
Depends R (>= 3.0.1)
LazyData true
NeedsCompilation no
Author Yichuan Bai [aut, cre], Lynna Chu [aut]
Maintainer Yichuan Bai <ycbai@iastate.edu>
Repository CRAN
Date/Publication 2023-08-14 11:40:02 UTC

R topics documented:

example0 ................................................................. 2
getdis ................................................................. 2
rg.test ................................................................. 3
weiArith ................................................................. 5
weiGeo ................................................................. 6
weiMax ................................................................. 6

Index 8
example0

Description
These example contains a dataset, the label of the observations in the dataset, the distance matrix of the dataset using L2 distance, and the edge matrix generated by 5-MST.

Usage
example0

Format
An object of class list of length 4.

Details
data pooled dataset of two samples sampling from two different t-distributions.
label label of the observations. ‘sample 1’ denotes the observations in sample 1. ‘sample 2’ denotes the observations in sample 2.
distance the distance matrix of the pooled dataset using L2 distance.
edge edge matrix generated by 5-MST.

gedis

Description
This function returns the distance matrix using L2 distance.

Usage
gedis(y)

Arguments
y dataset of the pooled data

Value
A distance matrix based on the L2 distance.
Examples

```r
data(example0)
data = as.matrix(example0$data)  # pooled dataset
getdis(data)
```

---

**rg.test**  
*Robust graph-based two sample test*

**Description**

Performs robust graph-based two sample test.

**Usage**

```r
rg.test(data.X, data.Y, dis = NULL, E = NULL, n1, n2, k = 5, weigh.fun, perm.num = 0,
test.type = list("ori", "gen", "wei", "max"), progress_bar = FALSE)
```

**Arguments**

- `data.X`: a numeric matrix for observations in sample 1.
- `dis`: a distance matrix of the pooled dataset of sample 1 and sample 2. The indices of observations in sample 1 are from 1 to `n1` and indices of observations in sample 2 are from `1+n1` to `n1+n2` in the pooled dataset.
- `E`: an edge matrix representing a similarity graph. Each row represents an edge and records the indices of two ends of an edge in two columns. The indices of observations in sample 1 are from 1 to `n1` and indices of observations in sample 2 are from `1+n1` to `n1+n2`.
- `n1`: number of observations in sample 1.
- `n2`: number of observations in sample 2.
- `k`: parameter in K-MST, with default 5.
- `weigh.fun`: weighted function which returns weights of each edge and is a function of node degrees.
- `perm.num`: number of permutations used to calculate the p-value (default=1000). Use 0 for getting only the approximate p-value based on asymptotic theory.
- `test.type`: type of graph-based test. This must be a list containing elements chosen from "ori", "gen", "wei", and "max", with default `list("ori", "gen", "wei", "max")`. "ori" refers to robust original edge-count test, "gen" refers to robust generalized edge-count test, "wei" refers to robust weighted edge-count test and "max" refers to robust max-type edge-count tests.
- `progress_bar`: a logical evaluating to TRUE or FALSE indicating whether a progress bar of the permutation should be printed.
Details

The input should be one of the following:

1. datasets of the two samples;
2. the distance matrix of the pooled dataset;
3. the edge matrix generated from a similarity graph.

Typical usages are:

rg.test(data.X, data.Y, n1, n2, weigh.fun, ...)

rg.test(dis, n1, n2, weigh.fun, ...)

rg.test(E, n1, n2, weigh.fun, ...)

If the data matrices or the distance matrix are used, the similarity graph is generated using K-MST.

Value

A list containing the following components:

- **asy.ori.statistic**: the asymptotic test statistic using robust original graph-based test.
- **asy.ori.pval**: the asymptotic p-value using robust original graph-based test.
- **asy.gen.statistic**: the asymptotic test statistic using robust generalized graph-based test.
- **asy.gen.pval**: the asymptotic p-value using robust generalized graph-based test.
- **asy.wei.statistic**: the asymptotic test statistic using robust weighted graph-based test.
- **asy.wei.pval**: the asymptotic p-value using robust weighted graph-based test.
- **asy.max.statistic**: the asymptotic test statistic using robust max-type graph-based test.
- **asy.max.pval**: the asymptotic p-value using robust max-type graph-based test.
- **perm.ori.pval**: the p-value based on permutation using robust original graph-based test.
- **perm.gen.pval**: the p-value based on permutation using robust generalized graph-based test.
- **perm.wei.pval**: the p-value based on permutation using robust weighted graph-based test.
- **perm.max.pval**: the p-value based on permutation using robust max-type graph-based test.
## Simulated from Student's t-distribution.
## Observations for the two samples are from different distributions.

```r
data(example0)
data = as.matrix(example0$data) # pooled dataset
label = example0$label # label of observations
s1 = data[label == 'sample 1', ] # sample 1
s2 = data[label == 'sample 2', ] # sample 2
num1 = nrow(s1) # number of observations in sample 1
num2 = nrow(s2) # number of observations in sample 2

## Graph-based two sample test using data as input
rg.test(data.X = s1, data.Y = s2, n1 = num1, n2 = num2, k = 5, weigh.fun = weiMax, perm.num = 0)

## Graph-based two sample test using distance matrix as input
dist = example0$distance
rg.test(dis = dist, n1 = num1, n2 = num2, k = 5, weigh.fun = weiMax, perm.num = 0)

## Graph-based two sample test using edge matrix of the similarity graph as input
E = example0$edge
rg.test(E = E, n1 = num1, n2 = num2, weigh.fun = weiMax, perm.num = 0)
```

---

### weiArith

**Weighted function**

This weight function returns the inverse of the arithmetic average of the node degrees of an edge.

#### Usage

`weiArith(a, b)`

#### Arguments

- `a`: node degree of one end of an edge
- `b`: node degree of another end of an edge

#### Value

The weight uses the arithmetic average of the node degrees of an edge.

#### Examples

```r
# For an edge where one end has a node degree of 5
# another end has a node degree of 6
weiArith(6, 5)
```
### weiGeo

**Weighted function**

**Description**
This weight function returns the inverse of the geometric average of the node degrees of an edge.

**Usage**

```plaintext
weiGeo(a, b)
```

**Arguments**

- `a` node degree of one end of an edge
- `b` node degree of another end of an edge

**Value**

The weight uses the geometric average of the node degrees of an edge.

**Examples**

```plaintext
# For an edge where one end has a node degree of 5
# another end has a node degree of 6
weiGeo(6, 5)
```

---

### weiMax

**Weighted function**

**Description**
This weight function returns the inverse of the max node degree of an edge.

**Usage**

```plaintext
weiMax(a, b)
```

**Arguments**

- `a` node degree of one end of an edge
- `b` node degree of another end of an edge

**Value**

The weight uses the max node degrees of an edge.
Examples

    # For an edge where one end has a node degree of 5
    # another end has a node degree of 6
    weiMax(6, 5)
Index

* datasets
  example0, 2

example0, 2
getdis, 2
rg.test, 3
weiArith, 5
weiGeo, 6
weiMax, 6