Package 'GRelevance'

May 7, 2026

Title Graph-Based k-Sample Comparisons and Relevance Analysis in High Dimensions

Version 1.0

Imports mvtnorm,MASS,philentropy

Description
We propose two distribution-free test statistics based on between-sample edge counts and measure the degree of relevance by standardized counts. Users can set edge costs in the graph to compare the parameters of the distributions. Methods for comparing distributions are as described in: Xiaoping Shi (2021) <doi:10.48550/arXiv.2107.00728>.

Encoding UTF-8

RoxygenNote 7.2.0

Depends R (>= 2.10)

License MIT + file LICENSE

NeedsCompilation no

Author Xiaoping Shi [aut, cre] (ORCID: <https://orcid.org/0000-0001-7981-0708>)

Maintainer Xiaoping Shi <xiaoping.shi@ubc.ca>

Repository CRAN

Date/Publication 2023-02-22 15:10:12 UTC

Contents

compbypath *Basic description*

Description

Given the groups and the shortest Hamiltonian path, this function returns the number of edges that connect nodes between samples.

Usage

```
compbypath(G,re.path)
```

Arguments

G	a list of all groups
re.path	the shortest Hamiltonian path returned from the function Hpath

Value

the number of edges that connect nodes between samples

See Also

Hpath

Examples

```
d=100;n1=20;n2=30;n3=40;
N=n1+n2+n3
mu1=rep(0,d)
mu2=mu1
mu3=mu2+0.1
cov1=0.2^(abs(outer(1:d,1:d,"-")))
cov2=0.2^(abs(outer(1:d,1:d,"-")))
cov3=0.4^(abs(outer(1:d,1:d,"-")))
sam1=MASS::mvrnorm(n=n1,mu=mu1,Sigma=cov1)
sam2=MASS::mvrnorm(n=n2,mu=mu2,Sigma=cov2)
sam3=MASS::mvrnorm(n=n3,mu=mu3,Sigma=cov3)
Data=rbind(sam1,sam2,sam3)
Dist=philentropy::distance(Data, method = "euclidean")
Dist[lower.tri(Dist)] <- NA
Dist[diag(Dist)] <- NA
G=list()
G[[1]]=c(1:n1);G[[2]]=c((n1+1):(n1+n2));G[[3]]=c((n1+n2+1):(n1+n2+n3));
compbypath(G,Hpath(1,N,Dist))
```

Hpath *Basic description*

Description

Applies the path.kruskal function based on the nodes and edge.cost (sorts the weights from minimum to maximum). Given the starting node, ending node, and the distance matrix, this function returns the list of nodes of each edge from the shortest Hamiltonian path. We have the Hamiltonian path from path.kruskal

Usage

```
Hpath(n1,n2,mat)
```

Arguments

n1	starting node
n2	ending node
mat	distance matrix (distance type is determined by the reader)

Value

list of nodes of each edge from the shortest Hamiltonian path

See Also

path.kruskal

Examples

```
G=list()
set.seed(1)
n1=20;n2=40
N=n1+n2;
G[[1]]=c(1:n1);G[[2]]=c((n1+1):(n1+n2));
d=10
mu1=rep(0,d)
mu2=mu1+0.1
true.cov1=0.4^(abs(outer(1:d,1:d,"-")))
true.cov2=0.4^(abs(outer(1:d,1:d,"-")))
sam1=MASS::mvrnorm(n=n1,mu=mu1,Sigma=true.cov1)
sam2=MASS::mvrnorm(n=n2,mu=mu2,Sigma=true.cov2)
Data=rbind(sam1,sam2)
Dist=philentropy::distance(Data, method = "euclidean")
Dist[lower.tri(Dist)] <- NA
Dist[diag(Dist)] <- NA
Hpath(1,N,Dist)
```

Mpermut *Basic description*

Description

Given the groups and the observed statistic, this function returns the pvalue.

Usage

```
Mpermut(G,W,obs)
```

Arguments

G	a list of all groups
W	the weight matrix
obs	the observed statistic

Value

the pvalue

Examples

```
G=list()
set.seed(1)
n1=20;n2=40
N=n1+n2;
G[[1]]=c(1:n1);G[[2]]=c((n1+1):(n1+n2));
d=10
mu1=rep(0,d)
mu2=mu1+0.1
true.cov1=0.4^(abs(outer(1:d,1:d,"-")))
true.cov2=0.4^(abs(outer(1:d,1:d,"-")))
sam1=MASS::mvrnorm(n=n1,mu=mu1,Sigma=true.cov1)
sam2=MASS::mvrnorm(n=n2,mu=mu2,Sigma=true.cov2)
Data=rbind(sam1,sam2)
Dist=philentropy::distance(Data, method = "euclidean")
Dist[lower.tri(Dist)] <- NA
Dist[diag(Dist)] <- NA
counts=compbypath(G,Hpath(1,N,Dist))
W=Weight(G)
#W[i,j]=0 #if we donot consider this relevance between sample i and sample j
C=counts$EC
Z=(C-W$mean)*W$weight
obs=min(Z[!is.na(Z)])
Mpermut(G,W$weight,obs)
```

path.kruskal *Basic description*

Description

Calculates the shortest Hamiltonian path based on the sorted edge weights and the nodes

Usage

```
path.kruskal(nodes,edge_cost)
```

Arguments

nodes	sequence of nodes 1,...,n from the graph which is based on the high-dimensional data that is provided by the reader
edge_cost	sorted edge weights

Value

the shortest Hamiltonian path

See Also

Hpath

Examples

```
G=list()
set.seed(1)
n1=20;n2=40
N=n1+n2;
G[[1]]=c(1:n1);G[[2]]=c((n1+1):(n1+n2));
d=10
mu1=rep(0,d)
mu2=mu1+0.1
true.cov1=0.4^(abs(outer(1:d,1:d,"-")))
true.cov2=0.4^(abs(outer(1:d,1:d,"-")))
sam1=MASS::mvrnorm(n=n1,mu=mu1,Sigma=true.cov1)
sam2=MASS::mvrnorm(n=n2,mu=mu2,Sigma=true.cov2)
Data=rbind(sam1,sam2)
Dist=philentropy::distance(Data, method = "euclidean")
Dist[lower.tri(Dist)] <- NA
Dist[diag(Dist)] <- NA
mat=Dist
n1=1; n2=N; n0=n2-n1+1
edge.cost=matrix(NA,nrow=n0*(n0-1)/2,ncol=3)
temp=1;
for(i in n1:(n2-1))
 for(j in (i+1):(n2))
```

```
   {
   edge.cost[temp,3]=mat[i,j];edge.cost[temp,1]=i-n1+1;edge.cost[temp,2]=j-n1+1;temp=temp+1;}
edge.cost=edge.cost[sort.list(edge.cost[,3]), ]
path.kruskal(c(1:n0),edge.cost)
```

`Weight` *Basic description*

Description

Given the sampless, this function returns the mean and weight matrix.

Usage

```
Weight(G)
```

Arguments

`G`	a list of all groups

Value

the mean and weight matrix

Examples

```
G=list()
set.seed(1)
n1=20;n2=40
N=n1+n2;
G[[1]]=c(1:n1);G[[2]]=c((n1+1):(n1+n2));
Weight(G)
```

`Wpermut` *Basic description*

Description

Given the groups, the weight matrix and the observed statistic, this function returns the pvalue.

Usage

```
Wpermut(G,W,obs)
```

Arguments

G	a list of all groups
W	the weight matrix
obs	the observed statistic

Value

the pvalue

Examples

```
G=list()
set.seed(1)
n1=20;n2=40
N=n1+n2;
G[[1]]=c(1:n1);G[[2]]=c((n1+1):(n1+n2));
d=10
mu1=rep(0,d)
mu2=mu1+0.1
true.cov1=0.4^(abs(outer(1:d,1:d,"-")))
true.cov2=0.4^(abs(outer(1:d,1:d,"-")))
sam1=MASS::mvrnorm(n=n1,mu=mu1,Sigma=true.cov1)
sam2=MASS::mvrnorm(n=n2,mu=mu2,Sigma=true.cov2)
Data=rbind(sam1,sam2)
Dist=philentropy::distance(Data, method = "euclidean")
Dist[lower.tri(Dist)] <- NA
Dist[diag(Dist)] <- NA
counts=compbypath(G,Hpath(1,N,Dist))
W=Weight(G)
#W[i,j]=0 #if we donot consider this relevance between sample i and sample j
C=counts$EC
WC=W$weight*C
WS=sum(WC[!is.na(WC)])
Wpermut(G,W$weight,WS)
```

Index