

Package ‘EDOIF’

March 28, 2021

Title Empirical Distribution Ordering Inference Framework (EDOIF)

Version 0.1.3

Maintainer Chainarong Amornbunchornvej <grandca@gmail.com>

Description A non-parametric framework based on estimation statistics principle. Its main purpose is to infer orders of empirical distributions from different categories based on a probability of finding a value in one distribution that is greater than an expectation of another distribution. Given a set of ordered-pair of real-category values the framework is capable of 1) inferring orders of domination of categories and representing orders in the form of a graph; 2) estimating magnitude of difference between a pair of categories in forms of mean-difference confidence intervals; and 3) visualizing domination orders and magnitudes of difference of categories. The publication of this package is at Chainarong Amornbunchornvej, Navaporn Surasvadi, Anon Plangprasopchok, and Suttipong Thajchayapong (2020) <doi:10.1016/j.heliyon.2020.e05435>.

License BSD_3_clause + file LICENSE

URL <https://github.com/DarkEyes/EDOIF>

BugReports <https://github.com/DarkEyes/EDOIF/issues>

Language en-US

Encoding UTF-8

LazyData false

Depends R (>= 3.5.0), boot

Imports distr, igraph, ellipsis, simpleboot, ggplot2 (>= 3.0)

Suggests knitr, rmarkdown, markdown

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Chainarong Amornbunchornvej [aut, cre]
(<<https://orcid.org/0000-0003-3131-0370>>)

Repository CRAN

Date/Publication 2021-03-28 06:20:05 UTC

R topics documented:

bootDiffmeanFunc	2
checkSim3Res	3
EDOIF	4
getADJNetDen	6
getConfInv	7
getDominantRADJ	7
getiGraphNetDen	8
getiGraphOBJ	9
getMegDiffConfInv	10
getOrder	11
getttestDominantRADJ	11
getWilcoxDominantRADJ	12
meanBoot	13
plot.EDOIF	13
plotGraph	14
plotMeanCIs	15
plotMeanDiffCIs	16
print.EDOIF	17
SimMixDist	17
SimNonNormalDist	18
Index	20

bootDiffmeanFunc	<i>bootDiffmeanFunc function</i>
------------------	----------------------------------

Description

bootDiffmeanFunc is a support function for bootstrapping method. Its main task is to infer mean-difference confidence intervals of distributions for all categories except the first category in idx (idx[2],idx[3],...) minus a target category (idx[1]).

Usage

```
bootDiffmeanFunc(Group, Values, idx, reps, ci, methodType)
```

Arguments

Group	is a vector of categories of each real number in Values
Values	is a vector of real-number values
idx	is an order list of categories; idx[1] is a target category while others (idx[2],idx[3],...) are compared against idx[1] in order to compute mean-difference confidence intervals.
reps	is a number of time of sampling with replacement in a bootstrapping method.
ci	is a level of confidence interval inferred.

methodType is a type of method for inferring confidence intervals. It is a parameter of two.boot function of simpleboot package.

Value

This function returns a list of mean-difference confidence intervals of categories idx[2],idx[3],... minus category idx[1].

result a list of objects that contains mean-difference confidence intervals of pairs of distributions. It contains mean-difference confidence intervals of categories idx[2],idx[3],... minus category idx[1].

checkSim3Res	<i>checkSim3Res function</i>
--------------	------------------------------

Description

checkSim3Res is a support function for checking whether an adjacency matrix of inferred a dominant-distribution network adjMat is corrected w.r.t. generator SimNonNormalDist().

Usage

```
checkSim3Res(adjMat, flag = 0)
```

Arguments

adjMat is an adjacency matrix of inferred a dominant-distribution network.
 flag is a flag of matrix. It should be set only to shift the low of matrix for comparison.

Value

This function returns precision, recall, and F1-score of inferred adjacency matrix.

Examples

```
# Generate simulation data with 100 samples per categories
simData<-SimNonNormalDist(nInv=100)

# Performing ordering inference from simData
resultObj<-ED0IF(simData$Values,simData$Group)

# Compare the inferred adjacency matrix with the ground truth
checkSim3Res(adjMat=resultObj$adjMat)
```

Description

EDOIF is a non-parametric framework based on Estimation Statistics principle. Its main purpose is to infer orders of empirical distributions from different categories base on a probability of finding a value in one distribution that greater than the expectation of another distribution.

Given a set of ordered-pair of real-category values the framework is capable of 1) inferring orders of domination of categories and representing orders in the form of a graph; 2) estimating magnitude of difference between a pair of categories in forms of confidence intervals; and 3) visualizing domination orders and magnitudes of difference of categories.

Usage

```
EDOIF(Values, Group, bootT, alpha, methodType)
```

Arguments

Values	is a vector of real-number values
Group	is a vector of categories of each real number in Values
bootT	is a number of times of sample with replacement for bootstrapping. The default is 1000. It must be above zero
alpha	is a significance level using in both confidence intervals and ordering inference it has the range [0,1]. The default is 0.05.
methodType	is an option for bootstrapping methods:either "perc" or "bca". The "perc" is the default option.

Value

This class constructor returns an object of EDOIF class.

obj an object of EDOIF class that contains the results of ordering inference that can be print in text mode (print(obj)) or graphic mode (plot(obj)).

The obj consists of the following variables

Values, Group	The main inputs of the framework. They are the double and character vectors respectively.
bootT, alpha, methodType	The number of bootstrapping, significance level, and bootstrapping method parameters.
sortedGroupList	A list of names of categories ascendingly ordered by their means.
sortedmeanList	A list of means of categories that are ascendingly ordered.

MegDiffList[[i]]	Mean difference confidence intervals and related information of all categories that have higher means than sortedGroupList[i] category.
confInvsList[i,]	A mean confidence interval of sortedGroupList[i] category. confInvsList[i,1] is a lower bound and confInvsList[i,2] is an upper bound.
adjMat[i, j]	An element of adjacency matrix: one if sortedGroupList[j] category dominates sortedGroupList[i] using Mann-Whitney test, otherwise zero.
pValMat[i, j]	A p-value of Mann-Whitney test for adjMat[i,j].
adjDiffMat[i, j]	A lower bound of confidence interval of mean difference for sortedGroupList[j] minus sortedGroupList[i] using methodType bootstrap.
adjBootMat[i, j]	One if adjDiffMat[i,j] is positive, otherwise, zero.
netDen	A network density of dominant-distribution network derived from adjMat.
gObj	An object of iGraph of a dominant-distribution network.

Author(s)

Chainarong Amornbunchornvej, <chai@ieee.org>

See Also

Run vignette("EDOIF_demo", package = "EDOIF") in a terminal to learn more details about how to use our package.

Examples

```
# Generate simulation data
nInv<-100
initMean=10
stepMean=20
std=8
simData1<-c()
simData1$Values<-rnorm(nInv,mean=initMean,sd=std)
simData1$Group<-rep(c("C1"),times=nInv)
simData1$Values<-c(simData1$Values,rnorm(nInv,mean=initMean,sd=std) )
simData1$Group<-c(simData1$Group,rep(c("C2"),times=nInv))
simData1$Values<-c(simData1$Values,rnorm(nInv,mean=initMean+2*stepMean,sd=std) )
simData1$Group<-c(simData1$Group,rep(c("C3"),times=nInv) )
simData1$Values<-c(simData1$Values,rnorm(nInv,mean=initMean+3*stepMean,sd=std) )
simData1$Group<-c(simData1$Group, rep(c("C4"),times=nInv) )
simData1$Values<-c(simData1$Values,rnorm(nInv,mean=initMean+4*stepMean,sd=std) )
simData1$Group<-c(simData1$Group, rep(c("C5"),times=nInv) )

# Performing ordering inference from simData1

resultObj<-EDOIF(simData1$Values,simData1$Group)
```

```
# Print results in text mode
print(resultObj)

# Plot results in graphic mode

plot(resultObj)
```

getADJNetDen *getADJNetDen function*

Description

getADJNetDen is a support function for calculating a network density of a dominant-distribution network.

Usage

```
getADJNetDen(adjMat)
```

Arguments

adjMat is an adjacency matrix of a dominant-distribution network.

Value

This function returns a value of network density of of a dominant-distribution network for a given adjMat.

Examples

```
# Generate simulation data with 100 samples per categories

simData<-SimNonNormalDist(nInv=100)

# Performing ordering inference from simData

resultObj<-EDOIF(simData$Values,simData$Group)

# Get a network density of an adjacency matrix

getADJNetDen(adjMat=resultObj$adjMat)
```

getConfInv *getConfInv function*

Description

getConfInv is a support function for bootstrapping method. Its main purpose is to compute a mean confidence intervals of all distributions.

Usage

```
getConfInv(Values, Group, GroupList, bootT, alpha, methodType)
```

Arguments

Values	is a vector of real-number values
Group	is a vector of categories of each real number in Values
GroupList	is a list of names of categories ascendingly ordered by their means.
bootT	is a number of times of sample with replacement for bootstrapping. The default is 1000. It must be above zero
alpha	is a significance level using in both confidence intervals and ordering inference it has the range [0,1]. The default is 0.05.
methodType	is an option for bootstrapping methods:either "perc" or "bca". The "perc" is the default option.

Value

This function returns a list of mean confidence intervals.

```
confInvsList[i,]
                    The mean confidence interval of sortedGroupList[i] category. confInvsList[i,1]
                    is a lower bound and confInvsList[i,2] is an upper bound.
```

getDominantRADJ *getDominantRADJ function*

Description

getDominantRADJ is a support function for inferring a dominant-distribution network using mean-difference confidence intervals.

Usage

```
getDominantRADJ(MegDiffList, methodType)
```

Arguments

- `MegDiffList` is a list of objects that contains mean-difference confidence intervals inferred by `getMegDiffConfInv` function.
- `methodType` is an option for bootstrapping methods: either "perc" or "bca".

Value

This function returns an adjacency matrix of a dominant-distribution network `adjMat` and the corresponding lower-bound of mean difference CIs `adjDiffMat`.

- `adjDiffMat[i, j]` A lower bound of confidence interval of mean difference for `j` minus `i` using `methodType` bootstrap.
- `adjMat[i, j]` An element of adjacency matrix: One if `adjDiffMat[i, j]` is positive, otherwise, zero.

`getiGraphNetDen` *getiGraphNetDen* function

Description

`getiGraphNetDen` is a support function for calculating a network density of a dominant-distribution network.

Usage

```
getiGraphNetDen(g)
```

Arguments

- `g` is an object of `iGraph` class of a dominant-distribution network.

Value

This function returns a value of network density of of a dominant-distribution network for a given object `g`.

Examples

```
# Generate simulation data with 100 samples per categories
simData<-SimNonNormalDist(nInv=100)

# Performing ordering inference from simData
resultObj<-ED0IF(simData$Values,simData$Group)

# Get a network density of an iGraph object
```



```
getiGraphNetDen(g=resultObj$gObj)
```

getiGraphOBJ	<i>getiGraphOBJ function</i>
--------------	------------------------------

Description

getiGraphOBJ is a support function for converting a dominant-distribution network adjacency matrix to an iGraph object.

Usage

```
getiGraphOBJ(adjMat, sortedGroupList)
```

Arguments

adjMat is an adjacency matrix of a dominant-distribution network.

sortedGroupList is a list of names of categories ascendingly ordered by their means.

Value

This function returns an iGraph object of a dominant-distribution network for a given adjMat.

Examples

```
# Generate simulation data with 100 samples per categories
simData<-SimNonNormalDist(nInv=100)
# Performing ordering inference from simData
resultObj<-EDOIF(simData$Values,simData$Group)
# Get an iGraph object from an adjacency matrix
igraphObj<-getiGraphOBJ(adjMat=resultObj$adjMat,sortedGroupList=resultObj$sortedGroupList)
```

getMegDiffConfInv *getMegDiffConfInv function*

Description

getMegDiffConfInv is a support function for bootstrapping method. Its main purpose is to compute a mean-difference confidence intervals between all pair of distributions.

Usage

```
getMegDiffConfInv(Values, Group, GroupList, bootT, alpha, methodType)
```

Arguments

Values	is a vector of real-number values
Group	is a vector of categories of each real number in Values
GroupList	is a list of names of categories ascendingly ordered by their means.
bootT	is a number of times of sample with replacement for bootstrapping. The default is 1000. It must be above zero
alpha	is a significance level using in both confidence intervals and ordering inference it has the range [0,1]. The default is 0.05.
methodType	is an option for bootstrapping methods:either "perc" or "bca". The "perc" is the default option.

Value

This function returns a list of mean-difference confidence intervals.

MegDiffList a list of objects that contains mean-difference confidence intervals of all possible pairs of distributions. It contains MegDiffList[[1]],...,MegDiffList[[length(GroupList)]].

The MegDiffList consists of the following variables

MegDiffList[[i]]

Mean-difference confidence intervals and related information of all categories that have higher means than sortedGroupList[i] category.

getOrder *getOrder function*

Description

getOrder is a support function for inferring a linear order of categories ascendingly sorted by their means.

Usage

```
getOrder(Values, Group)
```

Arguments

Values is a vector of real-number values
Group is a vector of categories of each real number in Values

Value

This function returns two lists: an order list of categories sortedGroupList and its corresponding list of means sortedmeanList.

sortedGroupList

The list of names of categories ascendingly ordered by their means.

sortedmeanList The list of means of categories that are ascendingly ordered.

Examples

```
# Generate simulation data  
  
simData<-SimNonNormalDist(nInv=100,noisePer=0.1)  
  
# Call the function to get the sorted lists  
getOrder(Values=simData$Values,Group=simData$Group)
```

getttestDominantRADJ *getttestDominantRADJ function*

Description

getttestDominantRADJ is a support function for inferring a dominant-distribution network using Student's t-test.

Usage

```
getttestDominantRADJ(Values, Group, GroupList, alpha)
```

Arguments

Values	is a vector of real-number values
Group	is a vector of categories of each real number in Values
GroupList	is a list of names of categories ascendingly ordered by their means.
alpha	is a significance level using in both confidence intervals and ordering inference it has the range [0,1].

Value

This function returns an adjacency matrix of a dominant-distribution network `adjMat` and the corresponding p-values of all category pairs.

<code>adjMat[i, j]</code>	An element of adjacency matrix: one if <code>GroupList[j]</code> category dominates <code>GroupList[i]</code> using Student's t-test, otherwise zero.
<code>pValMat[i, j]</code>	A p-value of Student's t-test for <code>adjMat[i, j]</code> .

`getWilcoxDominantRADJ` *getWilcoxDominantRADJ function*

Description

`getWilcoxDominantRADJ` is a support function for inferring a dominant-distribution network using Mann-Whitney (Wilcoxon) Test.

Usage

```
getWilcoxDominantRADJ(Values, Group, GroupList, alpha)
```

Arguments

Values	is a vector of real-number values
Group	is a vector of categories of each real number in Values
GroupList	is a list of names of categories ascendingly ordered by their means.
alpha	is a significance level using in both confidence intervals and ordering inference it has the range [0,1].

Value

This function returns an adjacency matrix of a dominant-distribution network `adjMat`. and the corresponding p-values of all category pairs.

<code>adjMat[i, j]</code>	An element of adjacency matrix: one if <code>GroupList[j]</code> category dominates <code>GroupList[i]</code> using Mann-Whitney test, otherwise zero.
<code>pValMat[i, j]</code>	A p-value of Mann-Whitney test for <code>adjMat[i, j]</code> .

meanBoot	<i>meanBoot function</i>
----------	--------------------------

Description

meanBoot is a support function for bootstrapping method. Its main purpose is to compute a mean of a given samples from data selected by indices.

Usage

```
meanBoot(data, indices)
```

Arguments

data	is a vector of real-number values
indices	is a vector of TRUE/FALSE indices. It allows boot to select samples.

Value

This function returns a mean of values in data that have values TRUE within indices.

plot.EDOIF	<i>plot.EDOIF function</i>
------------	----------------------------

Description

plot.EDOIF is a support function for printing all plots of EDOIF framework: dominant-distribution network plot, mean CI plot, and mean-difference CI plot.

Usage

```
## S3 method for class 'EDOIF'
plot(x, ..., NList, options, fontSize)
```

Arguments

x	is an object of EDOIF class that contains the results of ordering inference.
...	Signature for S3 generic function.
NList	is a list of based categories users want to have in mean-difference CI plot.
options	is an option of reporting EDOIF plot(s): 0 for reporting all plots, 1 for mean-difference CI plot, 2 for mean CI plot, and 3 for dominant-distribution network plot.
fontSize	is a font size of text for all plots.

Examples

```
# Generate simulation data with 100 samples per categories

simData<-SimNonNormalDist(nInv=100)

# Performing ordering inference from simData

resultObj<-EDOIF(simData$Values,simData$Group)

# Plot results in graphic mode

plot(resultObj)
```

plotGraph

plotGraph function

Description

plotGraph is a support function for plotting a dominant-distribution network from an adjacency matrix.

Usage

```
plotGraph(obj, rankFlag = TRUE)
```

Arguments

obj is an object of EDOIF class that contains the results of ordering inference.
rankFlag is an option for including ranks of categories with in the plot: default is TRUE for including ranks.

Value

This function returns a list of an object of iGraph for a dominant-distribution network and its plot variable.

graphVar An object of iGraph for a dominant-distribution network

Examples

```
# Generate simulation data with 100 samples per categories

simData<-SimNonNormalDist(nInv=100)

# Performing ordering inference from simData

resultObj<-EDOIF(simData$Values,simData$Group)
```

```
# Plot a dominant-distribution network and return a list of an iGraph object  
iGraphList<-plotGraph(obj=resultObj)
```

plotMeanCIs *plotMeanCIs function*

Description

plotMeanCIs is a support function for plotting mean confidence intervals.

Usage

```
plotMeanCIs(obj, fontSize = 15, rankFlag = TRUE)
```

Arguments

obj	is an object of EDOIF class that contains the results of ordering inference.
fontSize	is a font size of text for all plots.
rankFlag	is an option for including ranks of categories with in the plot: default is TRUE for including ranks.

Value

This function returns a list of an object of ggplot class.

pMeanCI	An object of ggplot class containing the plot of mean confidence intervals
---------	--

Examples

```
# Generate simulation data with 100 samples per categories  
simData<-SimNonNormalDist(nInv=100)  
  
# Performing ordering inference from simData  
resultObj<-EDOIF(simData$Values,simData$Group)  
  
# Get a list of ggplot object of mean confidence intervals  
ggplotList<-plotMeanCIs(obj=resultObj)  
  
# Plot mean confidence intervals  
plot(ggplotList$pMeanCI)
```

`plotMeanDiffCIs` *plotMeanDiffCIs* function

Description

`plotMeanDiffCIs` is a support function for plotting difference-mean confidence intervals.

Usage

```
plotMeanDiffCIs(obj, NList, fontSize = 15, rankFlag = TRUE)
```

Arguments

<code>obj</code>	is an object of EDOIF class that contains the results of ordering inference.
<code>NList</code>	is a list of based categories users want to have in mean-difference CI plot.
<code>fontSize</code>	is a font size of text for all plots.
<code>rankFlag</code>	is an option for including ranks of categories with in the plot: default is TRUE for including ranks.

Value

This function returns a list of an object of ggplot class.

<code>pDiffCI</code>	An object of ggplot class containing the plot of mean-difference confidence intervals
----------------------	---

Examples

```
# Generate simulation data with 100 samples per categories
simData<-SimNonNormalDist(nInv=100)

# Performing ordering inference from simData
resultObj<-EDOIF(simData$Values,simData$Group)

# Get a list of ggplot object of mean-difference confidence intervals
ggplotList<-plotMeanDiffCIs(obj=resultObj)

# Plot mean-difference confidence intervals
plot(ggplotList$pDiffCI)
```

print.EDOIF	<i>print.EDOIF function</i>
-------------	-----------------------------

Description

print.EDOIF is a support function for printing results of ordering inference in text.

Usage

```
## S3 method for class 'EDOIF'  
print(x, ...)
```

Arguments

x is an object of EDOIF class that contains the results of ordering inference.
... Signature for S3 generic function.

Examples

```
# Generate simulation data with 100 samples per categories  
  
simData<-SimNonNormalDist(nInv=100)  
  
# Performing ordering inference from simData  
  
resultObj<-EDOIF(simData$Values,simData$Group)  
  
# Print results in text mode  
  
print(resultObj)
```

SimMixDist	<i>SimMixDist function</i>
------------	----------------------------

Description

SimMixDist is a support function for generating samples from mixture distribution. The main purpose of this function is to generate samples from non-normal distribution.

Usage

```
SimMixDist(nInv, mean, std, p1, p2)
```

Arguments

nInv	is a number of samples the function will generate.
mean	is a mean of a normal distribution part of mixture distribution.
std	is a standard deviation of a normal distribution part of mixture distribution.
p1	is a ratio of a normal distribution within a mixture distribution.
p2	is a ratio of a Cauchy distribution within a mixture distribution.

Value

This function returns a list of samples V generated by a mixture distribution.

Examples

```
# Generate simulation data with 100 samples with a mixture distribution
# The distribution consist of the following distributions:
# 1) 10% of uniform distribution range [-400,400];
# 2) 50% of normal distribution with mean = 40 and std =8; and
# 3) 40% of Cauchy distribution with location= 45 and scale = 2.

V<-SimMixDist(nInv=100,mean=40,std=8,p1=0.1,p2=0.5)
```

SimNonNormalDist *SimNonNormalDist function*

Description

SimNonNormalDist is a support function for generating samples from mixture distribution. There are five categories. Each categories has nInv samples. Categories C1,C2,C3, and C4 are dominated by C5 but none of them dominate each other.

Usage

```
SimNonNormalDist(nInv, noisePer)
```

Arguments

nInv	is a number of samples the function will generate for each category.
noisePer	is ratio of uniform distribution within a mixture distribution. It is considered as a uniform noise that make an approach to hardly distinguish whether one distribution dominates another.

Details

The main purpose of this function is to generate samples that contains domination relation among categories.

Value

This function returns a list of samples `Values` and their category `Group` generated by a mixture distribution.

`Values` A vector of samples generated by a mixture distribution.

`Group` A list of categories associated with `Values`.

`V1, ..., V5` Lists of sample vectors separated by categories.

Examples

```
# Generate simulation data with 100 samples per categories with 10% of uniform noise  
  
simData<-SimNonNormalDist(nInv=100,noisePer=0.1)
```

Index

[bootDiffmeanFunc](#), 2

[checkSim3Res](#), 3

[EDOIF](#), 4

[getADJNetDen](#), 6

[getConfInv](#), 7

[getDominantRADJ](#), 7

[getiGraphNetDen](#), 8

[getiGraphOBJ](#), 9

[getMegDiffConfInv](#), 10

[getOrder](#), 11

[gettttestDominantRADJ](#), 11

[getWilcoxDominantRADJ](#), 12

[meanBoot](#), 13

[plot.EDOIF](#), 13

[plotGraph](#), 14

[plotMeanCIs](#), 15

[plotMeanDiffCIs](#), 16

[print.EDOIF](#), 17

[SimMixDist](#), 17

[SimNonNormalDist](#), 18