Package 'HHG'

July 14, 2015

Type Package

Title Heller-Heller-Gorfine Tests of Independence and Equality of Distributions

Version 1.5.1

Date 2015-07-13

Author Barak Brill & Shachar Kaufman, based in part on an earlier implementation by Ruth Heller and Yair Heller.

Maintainer Barak Brill <barakbri@mail.tau.ac.il>

Depends R (>= 3.1.0)

Suggests MASS,knitr

Description Heller-Heller-Gorfine (HHG) tests are a set of powerful statistical tests of multivariate k-sample homogeneity and independence. For the univariate case, the package also offers implementations of the MinP DDP and MinP ADP tests, which are consistent against all continuous alternatives but are distribution-free, and are thus much faster to apply.

License GPL-2

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-07-14 21:44:05

R topics documented:

`HHG-package` *Heller-Heller-Gorfine (HHG) Tests of Independence and Equality of Distributions*

Description

This R package implements the permutation test of independnece between two random vectors of arbitrary dimensions, and equality of two or more multivariate distributions, introduced in Heller et al. (2013), as well as the distribution-free tests of independence and equality of distribution between two univariate random variables introduced in Heller et al. (2014).

Details

Package:	HHG
Type:	Package
Version:	1.5.1
Date:	2015-07-13
License:	GPL-2

The package contains five major functions:

`hhg.test` - the permutation test for independence of two multivariate (or univariate) vectors.

`hhg.test.k.sample` - the permutation test for equality of a multivariate (or univariate) distribution across K groups.

`hhg.test.2.sample` - the permutaiton test for equality of a multivariate (or univariate) distribution across 2 groups.

`hhg.univariate.ind.combined.test` - the distribution-free test for independence of two univariate random variables.

`hhg.univariate.ks.combined.test` - the distribution-free test for equality of a univariate distribution across K groups.

See `vignette('HHG')` for additional information.

Author(s)

Barak Brill & Shachar Kaufman, based in part on an earlier implementation of the original HHG test by Ruth Heller <ruheller@post.tau.ac.il> and Yair Heller <heller.yair@gmail.com>. Maintainer:

Barak Brill <barakbri@mail.tau.ac.il>

References

Heller, R., Heller, Y., and Gorfine, M. (2013). A consistent multivariate test of association based on ranks of distances. *Biometrika*, 100(2), 503-510.

Heller, R., Heller, Y., Kaufman S., Brill B., and Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:

# Some examples, for more see the vignette('HHG') and specific help pages

########################################
#1. Univariate Independence Example
########################################

N = 30
data = hhg.example.datagen(N, 'Parabola')
X = data[1,]
Y = data[2,]
plot(X,Y)

#Option 1: Perform the ADP combined test
#using partitions sizes up to 4. see documentation for other parameters of the combined test
#(it is recommended to use mmax >= 4, or the default parameter for large data sets)
combined = hhg.univariate.ind.combined.test(X,Y,nr.perm = 200,mmax=4)
combined


#Option 2: Perform the hhg test:

## Compute distance matrices, on which the HHG test will be based
Dx = as.matrix(dist((X), diag = TRUE, upper = TRUE))
Dy = as.matrix(dist((Y), diag = TRUE, upper = TRUE))

hhg = hhg.test(Dx, Dy, nr.perm = 1000)

hhg

########################################
#2. Univariate K-Sample Example
########################################

N0=50
N1=50
X = c(c(rnorm(N0/2,-2,0.7),rnorm(N0/2,2,0.7)),c(rnorm(N1/2,-1.5,0.5),rnorm(N1/2,1.5,0.5)))
Y = (c(rep(0,N0),rep(1,N1)))
#plot the two distributions by group index (0 or 1)
```

```
plot(Y,X)


#Option 1: Perform the Sm combined test


combined.test = hhg.univariate.ks.combined.test(X,Y)
combined.test


#Option 2: Perform the hhg K-sample test:


Dx = as.matrix(dist(X, diag = TRUE, upper = TRUE))

hhg = hhg.test.k.sample(Dx, Y, nr.perm = 1000)

hhg


########################################
#3. Multivariate Independence Example:
########################################

n=30 #number of samples
dimensions_x=5 #dimension of X matrix
dimensions_y=5 #dimension of Y matrix
X=matrix(rnorm(n*dimensions_x,mean = 0, sd = 1),nrow = n,ncol = dimensions_x) #generate noise
Y=matrix(rnorm(n*dimensions_y,mean =0, sd = 3),nrow = n,ncol = dimensions_y)

Y[,1] = Y[,1] + X[,1] + 4*(X[,1])^2 #add in the relations
Y[,2] = Y[,2] + X[,2] + 4*(X[,2])^2

#compute the distance matrix between observations.
#User may use other distance metrics.
Dx = as.matrix(dist((X)), diag = TRUE, upper = TRUE)
Dy = as.matrix(dist((Y)), diag = TRUE, upper = TRUE)

#run test
hhg = hhg.test(Dx, Dy, nr.perm = 1000)

hhg


########################################
#4. Multivariate K-Sample Example
########################################

#multivariate k-sample, with k=3 groups
n=100 #number of samples in each group
x1 = matrix(rnorm(2*n),ncol = 2) #group 1
x2 = matrix(rnorm(2*n),ncol = 2) #group 2
x2[,2] = 1*x2[,1] + x2[,2]
```

```
x3 = matrix(rnorm(2*n),ncol = 2) #group 3
x3[,2] = -1*x3[,1] + x3[,2]
x= rbind(x1,x2,x3)
y=c(rep(0,n),rep(1,n),rep(2,n)) #group numbers, starting from 0 to k-1

plot(x[,1],x[,2],col = y+1,xlab = 'first component of X',ylab = 'second component of X',
    main = 'Multivariate K-Sample Example with K=3 \n Groups Marked by Different Colors')

Dx = as.matrix(dist(x, diag = TRUE, upper = TRUE)) #distance matrix

hhg = hhg.test.k.sample(Dx, y, nr.perm = 1000)

hhg


## End(Not run)
```

HHG *Heller-Heller-Gorfine Tests of Independence and Equality of Distributions*

Description

These functions perform Heller-Heller-Gorfine (HHG) tests. Implemented are tests of independence between two random vectors (x and y) and tests of equality of 2 or more multivariate distributions.

Usage

```
hhg.test(Dx, Dy, ties = T, w.sum = 0, w.max = 2, nr.perm = 10000,
  is.sequential = F, seq.total.nr.tests = 1,
  seq.alpha.hyp = NULL, seq.alpha0 = NULL, seq.beta0 = NULL, seq.eps = NULL,
  nr.threads = 0, tables.wanted = F, perm.stats.wanted = F)

hhg.test.k.sample(Dx, y, w.sum = 0, w.max = 2, nr.perm = 10000,
  is.sequential = F, seq.total.nr.tests = 1,
  seq.alpha.hyp = NULL, seq.alpha0 = NULL, seq.beta0 = NULL, seq.eps = NULL,
  nr.threads = 0, tables.wanted = F, perm.stats.wanted = F)

hhg.test.2.sample(Dx, y, w.sum = 0, w.max = 2, nr.perm = 10000,
  is.sequential = F, seq.total.nr.tests = 1,
  seq.alpha.hyp = NULL, seq.alpha0 = NULL, seq.beta0 = NULL, seq.eps = NULL,
  nr.threads = 0, tables.wanted = F, perm.stats.wanted = F)
```

Arguments

Dx	a symmetric matrix of doubles, where element [i, j] is a norm-based distance between the i'th and j'th x samples.

`Dy` same as `Dx`, but for distances between y's (the user may choose any norm when computing `Dx`, `Dy`).

`y` a numeric or factor vector, whose values or levels are in (0, 1, ..., `K` - 1), for performing K-sample tests (including `K` = 2).

`ties` a boolean specifying whether ties in `Dx` and/or `Dy` exist and are to be properly handled (requires more computation).

`w.sum` minimum expected frequency taken into account when computing the `sum.chisq` statistic (must be non-negative, contribution of tables having cells with smaller values will be truncated to zero).

`w.max` minimum expected frequency taken into account when computing the `max.chisq` statistic (must be non-negative, contribution of tables having cells with smaller values will be truncated to zero).

`nr.perm` number of permutations from which a p-value is to be estimated (must be non-negative). Can be specified as zero if only the observed statistics are wanted, without p-values. The actual number of permutations used may be slightly larger when using multiple processing cores. A Wald sequential probability ratio test is optionally implemented, which may push the p-value to 1 and stop permuting if it becomes clear that it is going to be high. See Details below.

`is.sequential` boolean flag specifying whether Wald's sequential test is desired (see Details), otherwise a simple Monte-Carlo computation of `nr.perm` permutations is performed. When this argument is `TRUE`, either `seq.total.nr.tests` or (`seq.alpha.hyp`, `seq.alpha0`, `seq.beta0`, `seq.eps`) must be supplied by the user.

`seq.total.nr.tests`
the total number of hypotheses in the family of hypotheses simultaneously tested. When this optional argument is supplied, it is used to derive default values for the parameters of the Wald sequential test. The default derivation is done assuming a nominal `0.05` FDR level, and sets:
`seq.alpha.hyp = 0.05 / max(1, log(seq.total.nr.tests))`, `seq.alpha0 = 0.05`, `seq.beta0 = min(0.01, 0.05 / seq.total.nr.tests)`, `seq.eps = 0.01`. Alternatively, one can specify their own values for these parameters using the following arguments.

`seq.alpha.hyp` the nominal test size for this single test within the multiple testing procedure.

`seq.alpha0` the nominal test size for testing the side null hypothesis of p-value > `seq.alpha.hyp`.

`seq.beta0` one minus the power for testing the side null hypothesis of p-value > `seq.alpha.hyp`.

`seq.eps` approximation margin around `seq.alpha.hyp` that defines the p-value regions for the side null `p > seq.alpha.hyp * (1 + seq.eps)` and side alternative `p < seq.alpha.hyp * (1 - seq.eps)`.

`nr.threads` number of processing cores to use for p-value permutation. If left as zero, will try to use all available cores.

`tables.wanted` boolean flag determining whether to output detailed local 2x2 contingency tables.

`perm.stats.wanted`
boolean flag determining whether to output statistics values computed for all permutations (representing null distributions).

Details

The HHG test (Heller et al., 2013) is a powerful nonparametric test for association (or, alternatively, independence) between two random vectors (say, x and y) of arbitrary dimensions. It is consistent against virtually any form of dependence, and has been shown to offer significantly more power than alternative approaches in the face of simple, and, more importantly, complex high-dimensional dependence. The test relies on norm-based distance metrics in x and (separately) in y. The choice of metric for either variable is up to the user (e.g. Euclidean, Mahalanobis, Manhattan, or whatever is appropriate for the data). The general implementation in hhg.test takes the distance matrices computed on an observed sample, Dx and Dy, and starts form there.

hhg.test.k.sample and hhg.test.2.sample are optimized special cases of the general test, where y is a partition of samples in x to K or 2 groups, respectively.

When enabled by `is.sequential`, Wald's sequential test is implemented as suggested by Fay et al. (2007) in order to reduce the `O(nr.perm * n^2 * log(n))` computational compelxity of the permutation test to a more managable size. Especially when faced with high multiplicity, say `M` simultaneous tests, the necessary number of iterations may be quite large. For example, if it is desired to control the family-wise error rate (FWER) at level `alpha` using the Bonferroni correction, one needs a p-value of `alpha / M` to establish significance. This seems to suggest that the minimum number of permutations required is `nr.perm = M / alpha`. However, if it becomes clear after a smaller number of permutations that the null cannot be rejected, no additional permutations are needed, and the p-value can be conservatively estimated as 1. Often, only a handful of hypotheses in a family are expected to be non-null. In this case the number of permutations for testing all hypotheses using Wald's procedure is expected to be much lower than the full `M^2 / alpha`.

The target significance level of the sequential test is specified in the argument `seq.alpha.hyp`. It depends on the number of hypotheses `M`, and the type of multiplicity correction wanted. For the Bonferroni correction, the threshold is `alpha / M`. For the less conservative procedure of Benjamini & Hochberg (1995), it is `M1 * q / M`, where `q` is the desired false discovery rate (FDR), and `M1` is the (unknwon) number of true non-null hypotheses. Although `M1` is unknown, the investigator can sometimes estimate it conservatively (e.g., if at most 0.02 of the hypotheses are expected to be non-null, set `M1 = 0.02 * M`).

Value

Four statistics described in the original HHG paper are returned:

`sum.chisq` - sum of Pearson chi-squared statistics from the 2x2 contingency tables considered.

`sum.lr` - sum of liklihood ratio ("G statistic") values from the 2x2 tables.

`max.chisq` - maximum Pearson chi-squared statistic from any of the 2x2 tables.

`max.lr` - maximum G statistic from any of the 2x2 tables.

If `nr.perm > 0`, then estimated permutation p-values are returned as:

`perm.pval.hhg.sc`, `perm.pval.hhg.sl`, `perm.pval.hhg.mc`, `perm.pval.hhg.ml`

In order to give information that may help localize where in the support of the distributions of x and y there is departure from independence, if `tables.wanted` is true, the 2x2 tables themselves are provided in:

`extras.hhg.tbls`

This is a `n^2` by 4 matrix, whose columns are `A11`, `A12`, `A21`, `A22` as denoted in the original HHG paper. Row r of the matrix corresponds to `S_ij` in the same paper, where `i = 1 + floor((r - 1) / n)`,

and `j = 1 + ((r - 1) %% n)`. Since `S_ij` is never computed for `i == j`, rows `(0:(n - 1)) * n + (1:n)` contain `NA`s on purpose. The only other case where NAs will occur are for the 2 and K-sample tests, where only one table is given for any x-tied samples (the other tables at indices with the same x value are redundant).

Finally, as a means of estimating the null distributions of computed statistics, if `perm.stats.wanted` is true, the statistics computed for every permutation of the data performed during testing is outputted as:

`extras.perm.stats`

A data.frame with one variable per statistic and one sample per permutation.

`n` - The sample size

`y` - Group sizes for `hhg.test.2.sample` and `hhg.test.k.sample` tests.

`stat.type` - String holding the type of test used: `'hhg.test'`, `'hhg.test.2.sample'` or `'hhg.test.k.sample'`

Note

The computational complexity of the test is `n^2*log(n)`, where `n` is the number of samples. Thus, when the sample size is large, computing the test for many permutations may take a long time.

P-values are reproducible when setting set.seed(seed_value) before peforming the permutation test (also when computation is done in multithread).

Author(s)

Shachar Kaufman, based in part on an earlier version by Ruth Heller and Yair Heller.

References

Heller, R., Heller, Y., & Gorfine, M. (2013). A consistent multivariate test of association based on ranks of distances. *Biometrika*, 100(2), 503-510.

Fay, M., Kim., H., & Hachey, M. (2007). On using truncated sequential probability ratio test boundaries for Monte Carlo implementation of hypothesis tests. *Journal of Computational and Graphical Statistics*, 16(4), 946-967.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57, 289-300.

Examples

```
## Not run:
## 1. The test of independence

## 1.1. A non-null univariate example

## Generate some data from the Circle example
n = 50
X = hhg.example.datagen(n, 'Circle')
plot(X[1,], X[2,])

## Compute distance matrices, on which the HHG test will be based
```

```
Dx = as.matrix(dist((X[1,]), diag = TRUE, upper = TRUE))
Dy = as.matrix(dist((X[2,]), diag = TRUE, upper = TRUE))

## Compute HHG statistics, and p-values using 1000 random permutations
set.seed(1) #set the seed for the random permutations
hhg = hhg.test(Dx, Dy, nr.perm = 1000)

## Print the  statistics and their permutation p-value

hhg

## 1.2. A null univariate example

n = 50
X = hhg.example.datagen(n, '4indclouds')

Dx = as.matrix(dist((X[1,]), diag = TRUE, upper = TRUE))
Dy = as.matrix(dist((X[2,]), diag = TRUE, upper = TRUE))

set.seed(1) #set the seed for the random permutations
hhg = hhg.test(Dx, Dy, nr.perm = 1000)

hhg

## 1.3. A multivariate example
library(MASS)

n = 50
p = 5
x = t(mvrnorm(n, rep(0, p), diag(1, p)))
y = log(x ^ 2)
Dx = as.matrix(dist((t(x)), diag = TRUE, upper = TRUE))
Dy = as.matrix(dist((t(y)), diag = TRUE, upper = TRUE))

set.seed(1) #set the seed for the random permutations
hhg = hhg.test(Dx, Dy, nr.perm = 1000)

hhg

## 2. The k-sample test

n = 50
D = hhg.example.datagen(n, 'FourClassUniv')
Dx = as.matrix(dist(D$x, diag = TRUE, upper = TRUE))

set.seed(1) #set the seed for the random permutations
hhg = hhg.test.k.sample(Dx, D$y, nr.perm = 1000)

hhg

## End(Not run)
```

`hhg.example.datagen` *A set of example data generators used to demonstrate the HHG test.*

Description

Six examples (Circle, Diamond, Parabola, 2Parabolas, W, 4indclouds) are taken from Newton's introduction to the discussion of the Energy Test in The Annals of Applied Statistics (2009). These are simple univariate dependence structures (or independence, in the latter case) used to demonstrate the tests of independece. The remaining examples (TwoClassUniv, FourClassUniv, TwoClassMultiv) generate data suitable for demonstrating the k-sample test (and in particular, the two-sample test).

Usage

```
hhg.example.datagen(n, example)
```

Arguments

`n`	The desired sample size
`example`	The choice of example

Value

For `example` in {Circle, Diamond, Parabola, 2Parabolas, W, and 4indclouds}, a matrix of two rows is returned, one row per variable. Columns are i.i.d. samples. Given these data, we would like to test whether the two variables are statistically independent. Except for the 4indclouds case, all examples in fact have variables that are dependent. When `example` is one of {TwoClassUniv, FourClassUniv, TwoClassMultiv}, a list is returned with elements `x` and `y`. `y` is a vector with values either 0 or 1 (for TwoClassUniv and TwoClassMultiv) or in 0:3 for (for FourClassUniv). `x` is a real valued random variable (TwoClassUniv and FourClassUniv) or vector (TwoClassMultiv) which is not independent of `y`.

Author(s)

Shachar Kaufman and Ruth Heller

References

Newton, M.A. (2009). Introducing the discussion paper by Szekely and Rizzo. *Annals of applied statistics*, 3 (4), 1233-1235.

Examples

```
X = hhg.example.datagen(50, 'Diamond')
plot(X[1,], X[2,])

X = hhg.example.datagen(50, 'FourClassUniv')
plot(X)
```

`hhg.univariate.ind.combined.test` *The combined independence test statistics for all partition sizes*

Description

A Class of Consistent Univariate Distribution-Free Tests of Independence - function for computation of combined test

Usage

```
hhg.univariate.ind.combined.test(X,Y=NULL,NullTable=NULL,mmin=2,
  mmax=max(floor(sqrt(length(X))/2),2),variant='ADP',aggregation.type='sum',
  score.type='LikelihoodRatio', w.sum = 0, w.max = 2 ,combining.type='MinP',nr.perm=100)
```

Arguments

`X`	a numeric vector with observed `X` values, or the test statistic as output from `hhg.univariate.ind.stat`.
`Y`	a numeric vector with observed `Y` values. Leave as Null if the input to `X` is the test statistic.
`NullTable`	The null table of the statistic, which can be downloaded from the software website or computed by the function `hhg.univariate.ind.nulltable`.
`mmin`	The minimum partition size of the ranked observations, default value is 2.
`mmax`	The maximum partition size of the ranked observations, default value is half the square root of the number of observations. For a max aggregation.type, this parameter cannot be more than 2 for the ADP varaint and 4 for DDP variant
`variant`	a character string specifying the partition type, must be one of `"ADP"` (default) or `"DDP"`.
`aggregation.type`	a character string specifying the aggregation type, must be one of `"sum"` (default), `"max"`, or `"both"`.
`score.type`	a character string specifying the score type, must be one of `"LikelihoodRatio"` (default), `"Pearson"`, or `"both"`.
`w.sum`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Sum"` and `score.type="Pearson"`, default value 0.
`w.max`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Max"` and `score.type="Pearson"`, default value 2.
`combining.type`	a character string specifying the combining type, must be one of `"MinP"` (default), `"Fisher"`, or `"both"`.
`nr.perm`	The number of permutations for the null distribution. Ignored if `NullTable` is non-null.

Details

The test statistic and p-value of the recommended independence test between two univariate random variables in Heller et al. (2014). The default combining type in the minimum p-value, so the test statistic is the minimum p-value over the range of partition sizes m from `mmin` to `mmax`, where the p-value for a fixed partition size m is defined by the aggregation type and score type .

Value

Returns a `UnivariateStatistic` class object, with the following entries:

`MinP`	The test statistic when the combining type is `"MinP"`.
`MinP.pvalue`	The p-value when the combining type is `"MinP"`.
`MinP.m.chosen`	The partition size m for which the p-value was the smallest.
`Fisher`	The test statistic when the combining type is `"Fisher"`.
`Fisher.pvalue`	The p-value when the combining type is `"Fisher"`.
`m.stats`	The statistic for each m in the range `mmin` to `mmax`.
`pvalues.of.single.m`	The p-values for each m in the range `mmin` to `mmax`.
`generated_null_table`	The null table object. Null if `NullTable` is non-null.
`stat.type`	"Independence-Combined"
`variant`	a character string specifying the partition type used in the test, one of `"ADP"` or `"DDP"`.
`aggregation.type`	a character string specifying the aggregation type used in the , one of `"sum"` or `"max"`.
`score.type`	a character string specifying the score typeused in the test, one of `"LikelihoodRatio"` or `"Pearson"`.
`mmax`	The maximum partition size of the ranked observations used for MinP or Fisher test statistic.
`mmin`	The minimum partition size of the ranked observations used for MinP or Fisher test statistic.
`w.sum`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Sum"` and `score.type="Pearson"`.
`w.max`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Max"` and `score.type="Pearson"`.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:

N = 35
data = hhg.example.datagen(N, 'Parabola')
X = data[1,]
Y = data[2,]
plot(X,Y)

#I) Perform MinP & Fisher Tests - without existing null tables.
#Null tables are generated by the test function.
#using partitions sizes up to 5
results = hhg.univariate.ind.combined.test(X,Y,nr.perm = 100,mmax=5)
results


#The null table can then be accessed.
generated.null.table = results$generated_null_table


#II) Perform MinP & Fisher Tests - with existing null tables.

#create null table for aggregation by summation (on ADP), with partitions sizes up to 5:
ADP.null = hhg.univariate.ind.nulltable(N,mmax=5)

#create a null table, using aggregation by summation over DDP partitions,
#with partitions sizes up to 5, over Pearson scores,
#with 1000 bootstrap repetitions.
DDP.null = hhg.univariate.ind.nulltable(N,mmax = 5,variant = 'DDP',
score.type = 'Pearson', nr.replicates = 1000)

MinP.ADP.existing.null.table = hhg.univariate.ind.combined.test(X,Y, NullTable = ADP.null)
#Results
MinP.ADP.existing.null.table

#using the other null table (DDP variant, with pearson scores):
MinP.DDP.existing.null.table = hhg.univariate.ind.combined.test(X,Y, NullTable = DDP.null)

MinP.DDP.existing.null.table

# combined test can also be performed by using the test statistic.
ADP.statistic = hhg.univariate.ind.stat(X,Y,mmax=5)
MinP.using.statistic.result = hhg.univariate.ind.combined.test(ADP.statistic, NullTable = ADP.null)
# same result as above (as MinP.ADP.result.using.existing.null.table$MinP.pvalue)
MinP.using.statistic.result$MinP.pvalue


## End(Not run)
```

`hhg.univariate.ind.nulltable` *The independence test null tables for all partition sizes*

Description

Functions for creating null table objects, for the omnibus distribution-free test of independence between two univariate random variables, as described in Heller et al. (2014). To be used for the p-value computation, see examples in `hhg.univariate.ind.pvalue`.

Usage

```
hhg.univariate.ind.nulltable(size,mmin=2,mmax=max(floor(sqrt(size)/2),2),
variant='ADP',aggregation.type='sum',score.type='LikelihoodRatio',
w.sum = 0, w.max = 2,nr.replicates=1000,keep.simulation.data=F)
```

Arguments

`size`	The sample size
`mmin`	The minimum partition size of the ranked observations, default value is 2.
`mmax`	The maximum partition size of the ranked observations, default value is half the square root of the number of observations.
`variant`	a character string specifying the partition type, must be one of `"ADP"` (default) or `"DDP"`.
`aggregation.type`	a character string specifying the aggregation type, must be one of `"sum"` (default) or `"max"`.
`score.type`	a character string specifying the score type, must be one of `"LikelihoodRatio"` (default) or `"Pearson"`.
`w.sum`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Sum"` and `score.type="Pearson"`, default value 0.
`w.max`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Max"` and `score.type="Pearson"`, default value 2.
`nr.replicates`	The number of permutations for the null distribution.
`keep.simulation.data`	TRUE/FALSE. If TRUE, then in addition to the sorted statistics per column, the original matrix of size nr.replicates by mmax-mmin+1 is also stored.

Details

In order to compute the null distributions for a test statistic (with a specific aggregation and score type, and all partition sizes), the only necessary information is the sample size. The accuracy of the quantiles of the null distribution depend on the number of replicates used for constructing the null tables. The necessary accuracy depends on the threshold used for rejection of the null hypotheses.

Value

`m.stats` If keep.simulation.data= TRUE, `m.stats` a matrix with `nr.replicates` rows and `mmax-mmin+1` columns of null test statistics.

`univariate.object`
A useful format of the null tables for computing p-values efficiently.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:
#Testing for independance, sample size = 35
N=35

#null table for aggregation by summation (on ADP):
ADP.null = hhg.univariate.ind.nulltable(N)

#create a null table, using aggregation by summation over DDP partitions,
#with partitions sizes up to 5, over Pearson scores, with 1000 bootstrap repetitions.
DDP.null = hhg.univariate.ind.nulltable(N,mmax = 5,variant = 'DDP',
score.type = 'Pearson', nr.replicates = 1000)

## End(Not run)
```

---

hhg.univariate.ind.pvalue

The p-value computation for the test of independence using a fixed partition size

---

Description

The p-value computation for the distribution free test of independence between two univariate random variables of Heller et al. (2014) ,using a fixed partition size `m`.

Usage

```
hhg.univariate.ind.pvalue(statistic, NullTable, m=min(statistic$mmax,4))
```

Arguments

`statistic`	The value of the computed statistic by the function `hhg.univariate.ind.stat`. The statistic object includes the score type (one of `"LikelihoodRatio"` or `"Pearson"`), and the aggregation type (one of `"sum"` or `"max"`).
`NullTable`	The null table of the statistic, which can be downloaded from the software website (http://www.math.tau.ac.il/~ruheller/Software.html) or computed by the function `hhg.univariate.ind.nulltable`. See `vignette('HHG')` for a method of computing null tables on multiple cores.
`m`	The partition size.

Details

For the test statistic, the function extracts the fraction of observations in the null table that are at least as large as the test statistic, i.e. the p-value.

Value

The p-value.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:
N = 35
data = hhg.example.datagen(N, 'Parabola')
X = data[1,]
Y = data[2,]
plot(X,Y)


#I) Computing test statistics , with default parameters:

#statistic:
hhg.univariate.ADP.Likelihood.result = hhg.univariate.ind.stat(X,Y)
hhg.univariate.ADP.Likelihood.result

#null table:
ADP.null = hhg.univariate.ind.nulltable(N)
#pvalue:
hhg.univariate.ind.pvalue(hhg.univariate.ADP.Likelihood.result, ADP.null)
```

```
#II) Computing test statistics , with summation over Data Derived Partitions (DDP),
#using Pearson scores, and partition sizes up to 5:

#statistic:
hhg.univariate.DDP.Pearson.result = hhg.univariate.ind.stat(X,Y,variant = 'DDP',
  score.type = 'Pearson', mmax = 5)
hhg.univariate.DDP.Pearson.result

#null table:
DDP.null = hhg.univariate.ind.nulltable(N,mmax = 5,variant = 'DDP',
  score.type = 'Pearson', nr.replicates = 1000)

#pvalue , for different partition size:
hhg.univariate.ind.pvalue(hhg.univariate.DDP.Pearson.result, DDP.null, m =2)
hhg.univariate.ind.pvalue(hhg.univariate.DDP.Pearson.result, DDP.null, m =5)


## End(Not run)
```

`hhg.univariate.ind.stat` *The independence test statistics for all partition sizes*

Description

These statistics are used in the omnibus distribution-free test of independence between two univariate random variables, as described in Heller et al. (2014).

Usage

```
hhg.univariate.ind.stat(x, y, variant = 'ADP',aggregation.type='sum'
  ,score.type='LikelihoodRatio',mmax = max(floor(sqrt(length(x))/2),2),
  mmin =2 , w.sum = 0, w.max = 2)
```

Arguments

`x`	a numeric vector with observed `X` values (tied observations are broken at random).
`y`	a numeric vector with observed `Y` values (tied observations are broken at random).
`variant`	a character string specifying the partition type, must be one of `"ADP"` (default) or `"DDP"`.
`aggregation.type`	a character string specifying the aggregation type, must be one of `"sum"` (default), `"max"`, or `"both"`.

score.type	a character string specifying the score type, must be one of "LikelihoodRatio" (default), "Pearson", or "both".
mmax	The partition size of the ranked observations. The default size is half the square root of the number of observations
mmin	The partition size of the ranked observations. The default size is half the square root of the number of observations
w.sum	The minimum number of observations in a partition, only relevant for type="Independence", aggregation.type="Sum" and score.type="Pearson", default value 0.
w.max	The minimum number of observations in a partition, only relevant for type="Independence", aggregation.type="Max" and score.type="Pearson", default value 2.

Details

For each partition size $m = mmin, \ldots, mmax$, the function computes the scores in each of the paritions (according to score type), and aggregates all scores according to the aggregation type (see details in Heller et al. , 2014). If the score type is one of "LikelihoodRatio" or "Pearson", and the aggregation type is one of "sum" or "max", then the computed statistic will be in statistic, otherwise the computed statistics will be in the appropriate subset of sum.chisq, sum.lr, max.chisq, and max.lr. Note that if the variant is "ADP", all partition sizes are computed together in O(N^4), so the score computational complexity is O(N^4). For "DDP" and mmax>4,the score computational complexity is O(N^4)*(mmax-mmin+1).

Value

Returns a UnivariateStatistic class object, with the following entries:

statistic	The value of the computed statistic if the score type is one of "LikelihoodRatio" or "Pearson", and the aggregation type is one of "sum" or "max". One of sum.chisq, sum.lr, max.chisq, and max.lr.
sum.chisq	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the average over all Pearson chi-squared statistics from all the mXm contingency tables considered, divided by the total number of observations.
sum.lr	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the average over all LikelihoodRatio statistics from all the mXm contingency tables considered, divided by the total number of observations.
max.chisq	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the maximum over all Pearson chi-squared statistics from all the mXm contingency tables considered.
max.lr	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the maximum over all Pearson chi-squared statistics from all the mXm contingency tables considered.
type	"Independence"
stat.type	"Independence-Stat"
size	The sample size
score.type	The input score.type.

`aggregation.type`
: The input `aggregation.type`.

`mmin`
: The input `mmin`.

`mmax`
: The input `mmax`.

`additional`
: A vector with the input `w.sum` and `w.max`.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:
N = 35
data = hhg.example.datagen(N, 'Parabola')
X = data[1,]
Y = data[2,]
plot(X,Y)


#I) Computing test statistics , with default parameters(ADP statistic):

hhg.univariate.ADP.Likelihood.result = hhg.univariate.ind.stat(X,Y)

hhg.univariate.ADP.Likelihood.result

#II) Computing test statistics , with summation over Data Derived Partitions (DDP),
#using Pearson scores, and partition sizes up to 5:

hhg.univariate.DDP.Pearson.result = hhg.univariate.ind.stat(X,Y,variant = 'DDP',
  score.type = 'Pearson', mmax = 5)
hhg.univariate.DDP.Pearson.result

## End(Not run)
```

`hhg.univariate.ks.combined.test`
: *The combined K-sample test statistics and their p-values*

Description

Function for performing the combined omnibus distribution-free test of equality of distributions among K groups, as described in Heller et al. (2014). The test combines statistics from a range of partition sizes.

Usage

```
hhg.univariate.ks.combined.test(X,Y=NULL,NullTable=NULL,mmin=2,
  mmax=ifelse(is.null(Y),4,max(4,round(min(table(Y))/3))) ,
  aggregation.type='sum',score.type='LikelihoodRatio' ,combining.type='MinP',nr.perm=1000)
```

Arguments

`X`	A numeric vector of data values (tied observations are broken at random), or the test statistic as output from `hhg.univariate.ks.stat`.
`Y`	for k groups, a vector of integers with values `0:(k-1)` which specify the group each observation belongs to. Leave as Null if the input to `X` is the test statistic.
`NullTable`	The null table of the statistic, which can be downloaded from the software website or computed by the function `hhg.univariate.ks.nulltable`.
`mmin`	The minimum partition size of the ranked observations, default value is 2. Ignored if `NullTable` is non-null.
`mmax`	The maximum partition size of the ranked observations, default value is 1/3 the number of observations in the smallest group. Ignored if `NullTable` is non-null.
`aggregation.type`	a character string specifying the aggregation type, must be one of `"sum"` (default), or `"max"`. Ignored if `NullTable` is non-null or `X` is the test stiatistic.
`score.type`	a character string specifying the score type, must be one of `"LikelihoodRatio"` (default), or `"Pearson"`. Ignored if `NullTable` is non-null or `X` is the test stiatistic.
`combining.type`	a character string specifying the combining type, must be one of `"MinP"` (default), `"Fisher"`, or `"both"`.
`nr.perm`	The number of permutations for the null distribution. Ignored if `NullTable` is non-null.

Details

The test statistic and p-value of the recommended test for the K-sample problem in Heller et al. (2014). The default combining type in the minimum p-value, so the test statistic is the minimum p-value over the range of partition sizes m from `mmin` to `mmax`, where the p-value for a fixed partition size m is defined by the aggregation type and score type .

Value

Returns a `UnivariateStatistic` class object, with the following entries:

`MinP`	The test statistic when the combining type is `"MinP"`.

`MinP.pvalue` The p-value when the combining type is `"MinP"`.

`MinP.m.chosen` The partition size m for which the p-value was the smallest.

`Fisher` The test statistic when the combining type is `"Fisher"`.

`Fisher.pvalue` The p-value when the combining type is `"Fisher"`.

`m.stats` The statistic for each m in the range `mmin` to `mmax`.

`pvalues.of.single.m`
The p-values for each m in the range `mmin` to `mmax`.

`generated_null_table`
The null table object. Null if `NullTable` is non-null.

`stat.type` "KSample-Combined"

`aggregation.type`
a character string specifying the aggregation type used in the , one of `"sum"` or `"max"`.

`score.type` a character string specifying the score typeused in the test, one of `"LikelihoodRatio"` or `"Pearson"`.

`mmax` The maximum partition size of the ranked observations used for MinP or Fisher test statistic.

`mmin` The minimum partition size of the ranked observations used for MinP or Fisher test statistic.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:
#Two groups, each from a different normal mixture:
N0=30
N1=30
X = c(c(rnorm(N0/2,-2,0.7),rnorm(N0/2,2,0.7)),c(rnorm(N1/2,-1.5,0.5),rnorm(N1/2,1.5,0.5)))
Y = (c(rep(0,N0),rep(1,N1)))
plot(Y,X)

#I) Perform MinP & Fisher Tests - without existing null tables.
#Null tables are generated by the test function.

results = hhg.univariate.ks.combined.test(X,Y,nr.perm = 100)
results


#The null table can then be accessed.
```

```
generated.null.table = results$generated_null_table


#II)Perform MinP & Fisher Tests - with existing null tables.

#null table for aggregation by summation:
sum.nulltable = hhg.univariate.ks.nulltable(c(N0,N1), nr.replicates=1000)

MinP.Sm.existing.null.table = hhg.univariate.ks.combined.test(X,Y, NullTable = sum.nulltable)

#Results
MinP.Sm.existing.null.table

# combined test can also be performed by using the test statistic.
Sm.statistic = hhg.univariate.ks.stat(X,Y)
MinP.using.statistic = hhg.univariate.ks.combined.test(Sm.statistic, NullTable = sum.nulltable)
# same result as above
MinP.using.statistic$MinP.pvalue

#null table for aggregation by maximization:
max.nulltable = hhg.univariate.ks.nulltable(c(N0,N1), aggregation.type = 'max',
  score.type='LikelihoodRatio', mmin = 2, mmax = 10, nr.replicates = 100)

#combined test using both "MinP" and "Fisher":
MinPFisher.Mm.result = hhg.univariate.ks.combined.test(X,Y,NullTable =  max.nulltable ,
  combining.type = 'Both')
MinPFisher.Mm.result



## End(Not run)
```

`hhg.univariate.ks.nulltable` *The K-sample test null tables for all partition sizes*

Description

Functions for creating null table objects, for the omnibus distribution-free test of equality of distributions among K groups, as described in Heller et al. (2014). To be used for the p-value computation, see examples in `hhg.univariate.ks.pvalue`.

Usage

```
hhg.univariate.ks.nulltable(group.sizes,mmin=2,
mmax=max(4,round(min(group.sizes)/3)), aggregation.type='sum',
score.type='LikelihoodRatio',nr.replicates=1000,keep.simulation.data=F)
```

Arguments

group.sizes	the number of observations in each group.
mmin	The minimum partition size of the ranked observations, default value is 2.
mmax	The maximum partition size of the ranked observations, default value is 1/3 the number of observations in the smallest group.
aggregation.type	a character string specifying the aggregation type, must be one of "sum" (default) or "max".
score.type	a character string specifying the score type, must be one of "LikelihoodRatio" (default) or "Pearson".
nr.replicates	The number of permutations for the null distribution.
keep.simulation.data	TRUE/FALSE. If TRUE, then in addition to the sorted statistics per column, the original matrix of size nr.replicates by mmax-mmin+1 is also stored.

Details

In order to compute the null distributions for a test statistic (with a specific aggregation and score type, and all partition sizes), the only necessary information is the group sizes. The accuracy of the quantiles of the null distribution depend on the number of replicates used for constructing the null tables. The necessary accuracy depends on the threshold used for rejection of the null hypotheses.

Value

m.stats	If keep.simulation.data= TRUE, a matrix with nr.replicates rows and mmax-mmin+1 columns of null test statistics.
univariate.object	A useful format of the null tables for computing p-values efficiently.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:
#Testing for the difference between two groups, each from a normal mixture:
N0=30
N1=30

#null table for aggregation by summation:
sum.nulltable = hhg.univariate.ks.nulltable(c(N0,N1), nr.replicates=100)
#default nr. of replicates is 1000,
```

```
#but may take several seconds. For illustration only, we use 100 replicates,
#but it is highly recommended to use at least 1000 in practice.

#null table for aggregation by maximization:
max.nulltable = hhg.univariate.ks.nulltable(c(N0,N1), aggregation.type = 'max',
  score.type='LikelihoodRatio', mmin = 3, mmax = 5, nr.replicates = 100)
#default nr. of replicates is 1000, but may take several seconds. For illustration only,
#we use 100 replicates, but it is highly recommended to use at least 1000 in practice.


## End(Not run)
```

`hhg.univariate.ks.pvalue` *The p-value computation for the K-sample problem using a fixed partition size*

Description

The p-value computation for the K-sample test of Heller et al. (2014) using a fixed partition size `m`.

Usage

```
hhg.univariate.ks.pvalue(statistic, NullTable,m)
```

Arguments

`statistic`	The value of the computed statistic by the function `hhg.univariate.ks.stat`. The statistic object includes the score type (one of `"LikelihoodRatio"` or `"Pearson"`), and the aggregation type (one of `"sum"` or `"max"`).
`NullTable`	The null table of the statistic, which can be downloaded from the software website (http://www.math.tau.ac.il/~ruheller/Software.html) or computed by the function `hhg.univariate.ind.nulltable`. See `vignette('HHG')` for a method of computing null tables on multiple cores.
`m`	The partition size.

Details

For the test statistic, the function extracts the fraction of observations in the null table that are at least as large as the test statistic, i.e. the p-value.

Value

The p-value.

Author(s)

Barak Brill Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:

#Two groups, each from a different normal mixture:
N0=30
N1=30
X = c(c(rnorm(N0/2,-2,0.7),rnorm(N0/2,2,0.7)),c(rnorm(N1/2,-1.5,0.5),rnorm(N1/2,1.5,0.5)))
Y = (c(rep(0,N0),rep(1,N1)))
plot(Y,X)

#I)p-value for fixed partition size using the sum aggregation type
hhg.univariate.Sm.Likelihood.result = hhg.univariate.ks.stat(X,Y)
hhg.univariate.Sm.Likelihood.result


sum.nulltable = hhg.univariate.ks.nulltable(c(N0,N1), nr.replicates=100)
#default nr. of replicates is 1000, but may take several seconds.
#For illustration only, we use 100 replicates, but it is highly recommended
#to use at least 1000 in practice.

#p-value for m=4 (the default):
hhg.univariate.ks.pvalue(hhg.univariate.Sm.Likelihood.result, sum.nulltable, m=4)


#p-value for m=2:
hhg.univariate.ks.pvalue(hhg.univariate.Sm.Likelihood.result, sum.nulltable, m=2)

#II) p-value for fixed partition size using the max aggregation type

hhg.univariate.Mm.likelihood.result = hhg.univariate.ks.stat(X,Y,aggregation.type = 'max')

hhg.univariate.Mm.likelihood.result

max.nulltable = hhg.univariate.ks.nulltable(c(N0,N1), aggregation.type = 'max',
  score.type='LikelihoodRatio', mmin = 3, mmax = 5, nr.replicates = 100)
#default nr. of replicates is 1000, but may take several seconds.
#For illustration only, we use 100 replicates,
#but it is highly recommended to use at least 1000 in practice.

#p-value for m=3:
hhg.univariate.ks.pvalue(hhg.univariate.Mm.likelihood.result, max.nulltable ,m = 3)

#p-value for m=5:
```

```
hhg.univariate.ks.pvalue(hhg.univariate.Mm.likelihood.result, max.nulltable,m = 5)


## End(Not run)
```

`hhg.univariate.ks.stat`

The K-sample test statistics for all partition sizes

Description

These statistics are used in the omnibus distribution-free test of equality of distributions among K groups, as described in Heller et al. (2014).

Usage

```
hhg.univariate.ks.stat(x, y,aggregation.type='sum',score.type='LikelihoodRatio',
mmax = max(4,round(min(table(y))/3)),mmin=2)
```

Arguments

`x`	a numeric vector of data values. Tied observations are broken at random.
`y`	for `k` groups, a vector of integers with values `0:(k-1)` which specify the group each observation belongs to.
`aggregation.type`	a character string specifying the aggregation type, must be one of `"sum"` (default), `"max"`, or `"both"`.
`score.type`	a character string specifying the score type, must be one of `"LikelihoodRatio"` (default), `"Pearson"`, or `"both"`.
`mmax`	The maximum partition size of the ranked observations, default value is 1/3 the number of observations in the smallest group.
`mmin`	The minimum partition size of the ranked observations, default value is 2.

Details

For each partition size $m = mmin, \ldots, mmax$, the function computes the scores in each of the paritions (according to score type), and aggregates all scores according to the aggregation type (see details in Heller et al. , 2014). If the score type is one of `"LikelihoodRatio"` or `"Pearson"`, and the aggregation type is one of `"sum"` or `"max"`, then the computed statistic will be in `statistic`, otherwise the computed statistics will be in the appropriate subset of `sum.chisq`, `sum.lr`, `max.chisq`, and `max.lr`.

Value

Returns a `UnivariateStatistic` class object, with the following entries:

`statistic`	The value of the computed statistic if the score type is one of `"LikelihoodRatio"` or `"Pearson"`, and the aggregation type is one of `"sum"` or `"max"`. One of `sum.chisq`, `sum.lr`, `max.chisq`, and `max.lr`.
`sum.chisq`	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the average over all Pearson chi-squared statistics from all the KXm contingency tables considered, divided by the total number of observations.
`sum.lr`	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the average over all LikelihoodRatio statistics from all the KXm contingency tables considered, divided by the total number of observations.
`max.chisq`	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the maximum over all Pearson chi-squared statistics from all the KXm contingency tables considered.
`max.lr`	A vector of size $mmax - mmin + 1$, where the $m - mmin + 1$ entry is the maximum over all Pearson chi-squared statistics from all the KXm contingency tables considered.
`type`	"KSample".
`stat.type`	"KSample".
`size`	A vector of size K of the ordered group sample sizes.
`score.type`	The input `score.type`.
`aggregation.type`	The input `aggregation.type`.
`mmin`	The input `mmin`.
`mmax`	The input `mmax`.

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
#Example of computing the test statisics for data from a two-sample problem:

#Two groups, each from a different normal mixture:
X = c(c(rnorm(25,-2,0.7),rnorm(25,2,0.7)),c(rnorm(25,-1.5,0.5),rnorm(25,1.5,0.5)))
Y = (c(rep(0,50),rep(1,50)))
plot(Y,X)
```

```
#I) Computing test statistics , with default parameters:
hhg.univariate.Sm.Likelihood.result = hhg.univariate.ks.stat(X,Y)

hhg.univariate.Sm.Likelihood.result

#II) Computing test statistics , with max aggregation type:
hhg.univariate.Mm.likelihood.result = hhg.univariate.ks.stat(X,Y,aggregation.type = 'max')

hhg.univariate.Mm.likelihood.result
```

`hhg.univariate.nulltable.from.mstats`
Constructor of Distribution Free Null Table Using Existing Statistics

---

Description

This function converts null test statistics for different partition sizes into the null table object necessary for the computation of p-values efficiently.

Usage

```
hhg.univariate.nulltable.from.mstats(m.stats,minm,maxm,type,variant,size,score.type,
  aggregation.type, w.sum = 0, w.max = 2,keep.simulation.data=F)
```

Arguments

`m.stats`	A matrix with B rows and `maxm` - `minm`+1 columns, where each row contains the test statistics for partition sizes m from `minm` to `maxm` for the sample permutation of the input sample.
`minm`	The minimum partition size of the ranked observations, default value is 2.
`maxm`	The maximum partition size of the ranked observations.
`type`	A character string specifying the test type, must be one of `"KSample"`, `"Independence"`
`variant`	A character string specifying the partition type for the test of independence, must be one of `"ADP"`, `"DDP"` if `type="Independence"`. If `type="KSample"`, must be `"KSample-Variant"`.
`size`	The sample size if `type="Independence"`, and a vector of group sizes if `type="KSample"`.
`score.type`	a character string specifying the score type, must be one of `"LikelihoodRatio"`, or `"Pearson"`.
`aggregation.type`	a character string specifying the aggregation type, must be one of `"sum"`, or `"max"`.
`w.sum`	The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Sum"` and `score.type="Pearson"`, default value 0.

`w.max` The minimum number of observations in a partition, only relevant for `type="Independence"`, `aggregation.type="Max"` and `score.type="Pearson"`, default value 2.

`keep.simulation.data`
TRUE/FALSE.

Details

For finding multiple quantiles, the null table object is more efficient than a matrix of a matrix with B rows and `maxm - minm+1` columns, where each row contains the test statistics for partition sizes m from `minm` to `maxm` for the sample permutation of the input sample.

See `vignette('HHG')` for a section on how to use this function, for computing a null tables using multiple cores.

Value

`m.stats`The input `m.stats` if `keep.simulation.data=TRUE`

`univariate.object`A useful format of the null tables for computing p-values efficiently..

Author(s)

Barak Brill and Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., Brill B, & Gorfine, M. (2014). Consistent distribution-free K-sample and independence tests for univariate random variables *arXiv:1410.6758*.

Examples

```
## Not run:

# 1. Downloading a lookup table from site
# download from site http://www.math.tau.ac.il/~ruheller/Software.html
######################################################################
#using an already ready null table as object (for use in test functions)
#for example, ADP likelihood ratio statistics, for the independence problem, for sample size n=300
load('Object-ADP-n_300.Rdata') #=>null.table

#or using a matrix of statistics generated for the null distribution, to create your own table.
load('ADP-nullsim-n_300.Rdata') #=>mat
null.table = hhg.univariate.nulltable.from.mstats(m.stats = mat,minm = 2,
             maxm = 5,type = 'Independence', variant = 'ADP',size = 300,
             score.type = 'LikelihoodRatio',aggregation.type = 'sum')

# 2. generating an independence null table using multiple cores,
#and then compiling to object.
######################################################################
library(parallel)
library(doParallel)
library(foreach)
```

```
library(doRNG)

#generate an independence null table
nr.cores = 4 #this is computer dependent
n = 30 #size of independence problem
nr.reps.per.core = 25
mmax =5
score.type = 'LikelihoodRatio'
aggregation.type = 'sum'
variant = 'ADP'

#generating null table of size 4*25

#single core worker function
generate.null.distribution.statistic =function(){
  library(HHG)
  null.table = NULL
  for(i in 1:nr.reps.per.core){
    #note that the statistic is distribution free (based on ranks),
    #so creating a null table (for the null distribution)
    #is essentially permuting over the ranks
    statistic = hhg.univariate.ind.stat(1:n,sample(1:n),
                                        variant = variant,
                                        aggregation.type = aggregation.type,
                                        score.type = score.type,
                                        mmax = mmax)$statistic
    null.table=rbind(null.table, statistic)
  }
  rownames(null.table)=NULL
  return(null.table)
}

#parallelize over cores
cl = makeCluster(nr.cores)
registerDoParallel(cl)
res = foreach(core = 1:nr.cores, .combine = rbind, .packages = 'HHG',
              .export=c('variant','aggregation.type','score.type',
              'mmax','nr.reps.per.core','n'))
stopCluster(cl)

#the null table:
head(res)

#as object to be used:
null.table = hhg.univariate.nulltable.from.mstats(res,minm=2,
  maxm = mmax,type = 'Independence',
  variant = variant,size = n,score.type = score.type,
  aggregation.type = aggregation.type)

#using the null table, checking for dependence in a linear relation
x=rnorm(n)
y=x+rnorm(n)
ADP.test = hhg.univariate.ind.combined.test(x,y,null.table)
```

```
ADP.test$MinP.pvalue #pvalue


# 3. generating a k-sample null table using multiple cores
# and then compiling to object.
#####################################################################

library(parallel)
library(doParallel)
library(foreach)
library(doRNG)

#generate a k sample null table
nr.cores = 4 #this is computer dependent
n1 = 25 #size of first group
n2 = 25 #size of first group
nr.reps.per.core = 25
mmax =5
score.type = 'LikelihoodRatio'
aggregation.type = 'sum'

#generating null table of size 4*25

#single core worker function
generate.null.distribution.statistic =function(){
  library(HHG)
  null.table = NULL
  for(i in 1:nr.reps.per.core){
    #note that the statistic is distribution free (based on ranks),
    #so creating a null table (for the null distribution)
    #is essentially permuting over the ranks
    statistic = hhg.univariate.ks.stat(1:(n1+n2),sample(c(rep(0,n1),rep(1,n2))),
                                       aggregation.type = aggregation.type,
                                       score.type = score.type,
                                       mmax = mmax)$statistic
    null.table=rbind(null.table, statistic)
  }
  rownames(null.table)=NULL
  return(null.table)
}

#parallelize over cores
cl = makeCluster(nr.cores)
registerDoParallel(cl)
res = foreach(core = 1:nr.cores, .combine = rbind, .packages = 'HHG',
              .export=c('n1','n2','aggregation.type','score.type','mmax',
              'nr.reps.per.core'))
stopCluster(cl)

#the null table:
head(res)

#as object to be used:
```

```
null.table = hhg.univariate.nulltable.from.mstats(res,minm=2,
  maxm = mmax,type = 'KSample',
  variant = 'KSample-Variant',size = c(n1,n2),score.type = score.type,
  aggregation.type = aggregation.type)

#using the null table, checking for dependence in a case of two distinct samples
x=1:(n1+n2)
y=c(rep(0,n1),rep(1,n2))
Sm.test = hhg.univariate.ks.combined.test(x,y,null.table)
Sm.test$MinP.pvalue #pvalue

## End(Not run)
```

`print.HHG.Test.Result` *Print function for result of HHG tests*

Description

Print description of for result object of HHG tests

Usage

```
## S3 method for class 'HHG.Test.Result'
print(x, ...)
```

Arguments

`x`	result of `hhg.test`, `hhg.test.2.sample` or `hhg.test.k.sample`
`...`	Additional arguments can be sent to function. Currently not supported.

Details

Function prints description of results for the `hhg.test`, `hhg.test.2.sample` and `hhg.test.k.sample` functions. Displays: test statistics, pvalues (if permutations were performed) and description of sample size (also displays group sizes and equality of distribution tests).

Value

Does not return value. Only prints description of test statistic and results.

Author(s)

Barak Brill

Examples

```
#output for independence test
n = 50
X = hhg.example.datagen(n, '4indclouds')

Dx = as.matrix(dist((X[1,]), diag = TRUE, upper = TRUE))
Dy = as.matrix(dist((X[2,]), diag = TRUE, upper = TRUE))

hhg = hhg.test(Dx, Dy, nr.perm = 1000)

#output for k-sample test
n = 50
D = hhg.example.datagen(n, 'FourClassUniv')
Dx = as.matrix(dist(D$x, diag = TRUE, upper = TRUE))

hhg = hhg.test.k.sample(Dx, D$y, nr.perm = 1000)
```

---

print.UnivariateObject

Print function for Univariate Null Table Object

Description

Print description of univariate object.

Usage

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

Arguments

x	A univariate null table object, of type 'UnivariateObject'
...	Additional arguments can be sent to function. Currently not supported.

Details

Function prints description of a null table object, including sample size (for the independence hhg.univariate statsitics) or group sizes (for hhg.univarate statistics). Also prints statistic type in terms of `variant`, `aggregation.type` and `score.type`.

Value

Does not return value. Only prints description of null table object.

Author(s)

Barak Brill

Examples

```
#univariate objects are found inside null tables:
nt = hhg.univariate.ks.nulltable(group.sizes = c(20,20))

print(nt$univariate.object)
```

`print.UnivariateStatistic` *Print function for Univariate Statistic Test Object*

Description

Print description of univariate statistic result & test object.

Usage

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

Arguments

`x`	result of `hhg.univariate.ind.stat`, `hhg.univariate.ks.stat`, `hhg.univariate.ind.combined.test` or `hhg.univariate.ks.combined.test`
`...`	Additional arguments can be sent to function. Currently not supported.

Details

Function prints description of univariate test statistic and test results. Displays: test statistics, pvalues (for the combined test function), partition sizes and type of test statistic used in terms of `variant`, `aggregation.type` and `score.type`.

Value

Does not return value. Only prints description of test statistic and results.

Author(s)

Barak Brill

Examples

```
#generate statistics and test results, and print them
ind.stat = hhg.univariate.ind.stat(1:20,1:20,variant = 'ADP',
  aggregation.type = 'sum',score.type = 'both',mmax = 5)
print(ind.stat)

ks.stat = hhg.univariate.ks.stat(1:50,sample(c(rep(0,25),rep(1,25))),
```

```
  aggregation.type = 'both',score.type = 'both',mmax = 10)
print(ks.stat)

## Not run:

ind.combined = hhg.univariate.ind.combined.test(1:20,1:20,
  combining.type = 'Both',mmax = 5,nr.perm = 100)
print(ind.combined)


## End(Not run)

ks.combined = hhg.univariate.ks.combined.test(1:50,
  sample(c(rep(0,25),rep(1,25))),combining.type = 'Both')
print(ks.combined)
```

Yeast_hughes *Yeast gene expression data*

Description

The data is a small subset of the full dataset from Hughes et al. (2000).

Usage

```
data(Yeast_hughes)
```

Format

A data frame for 10 *S. cerevisae* (which is a common and well-studied species of Yeast) genes (stored in rows) and 300 expression observations (stored in columns). The 10 genes are those positioned 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 in chromosome I in the original data (see link below).

Source

Functional Discovery via a Compendium of Expression Profiles. Hughes et al., Cell, 2000.

The full data is publicly available at http://noble.gs.washington.edu/proj/microarray/Rosetta/.

References

Hughes et al.(2000). Functional Discovery via a Compendium of Expression Profiles. *Cell*, 102(Issue 1), P103-P126.

Examples

```
data(Yeast_hughes)
```

Index