Package 'HHG'

July 2, 2014

Type Package

Title Heller-Heller-Gorfine Tests of Independence

Version 1.4

Date 2014-05-05

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

Maintainer Shachar Kaufman <shachark@post.tau.ac.il>

Suggests MASS

Description Heller-Heller-Gorfine (HHG) tests are a set of powerful statistical tests of independence between two random vectors of arbitrary dimensions. For the case of testing independence between random variables (rather than vectors), the package also offers implementations of the DDP and ADP tests, which are consistent against all continuous alternatives but are distribution-free and thus much faster to apply.

License GPL-2

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-05-06 11:39:24

R topics documented:

`HHG-package` *Heller-Heller-Gorfine Tests of Independence*

Description

Heller-Heller-Gorfine (HHG) tests are a set of powerful statistical tests of independnece between two random vectors of arbitrary dimensions, given a finite sample. For testing independence between two scalar random variables, the package also contains implementations of the data-derived partitions (DDP) and all-data-partitions (ADP) tests, which are distribution-free and thus much faster to apply.

Details

Package:	HHG
Type:	Package
Version:	1.4
Date:	2014-05-05
License:	GPL-2

The package contains two major functions: `hhg.test`, which implements the HHG multivariate independence test described in Heller et al. (2013), and `xdp.test`, which implements the DDP and ADP univariate independence tests from Heller et al. (2014). In addition, in order to demonstrate the tests, the function `hhg.example.datagen` generates simulated data, and the dataset `hughes` contains some gene expression data.

Author(s)

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: Shachar Kaufman <shachark@post.tau.ac.il>

References

Heller, R., Heller, Y., & 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., & Gorfine, M. (2014). Consistent distribution-free tests of association between univariate random variables. *arXiv:1308.1559*.

Examples

```
# See examples in the documentation for hhg.test and xdp.test.
```

`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.

Usage

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

Arguments

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

Value

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.

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,])
```

`hhg.test` *Heller-Heller-Gorfine Tests of Independence*

Description

This function implements the Heller-Heller-Gorfine (HHG) test of independence between two random vectors (x and y) having arbitrary dimensions.

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)
```

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`).
`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.

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.

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.

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.

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.

Hall, P. & Tajvidi, N. (2002). Permutation tests for equality of distributions in high-dimensional settings. *Biometrika*, 89(2), 359-374.

Szekely, G., Rizzo, M., & Bakirov, N. (2007). Measuring and testing independence by correlation of distances. *The Annals of Statistics*, 35, 2769-2794.

Examples

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

## Print the sum-chisq statistic and its permutation p-value
hhg$sum.chisq
hhg$perm.pval.hhg.sc

## 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))

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

hhg$sum.chisq
hhg$perm.pval.hhg.sc

hhg$sum.lr
hhg$perm.pval.hhg.sl

## 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))

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

hhg$sum.chisq
hhg$perm.pval.hhg.sc
```

hughes — *Yeast gene expression data*

Description

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

Usage

```
data(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

Steuer, R., Kurths, J., Daub, C. O., Weise, J., & Selbig, J. (2002). The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl 2), S231-S240.

Examples

```
data(hughes)
```

xdp.test — *A Class of Consistent Univariate Distribution-Free Tests of Independence*

Description

This function computes test statistics for the tests of independence between two random variables (`X` and `Y`) presented in Heller et al. (2014). Several specialized variants of data-derived partitions (DDP) and all data partitions (ADP) tests are implemented.

Usage

```
xdp.test(x, y, variant = 'DDP', K = 3, correct.mi.bias = F, w.sum = 0, w.max = 2)
```

Arguments

`x`	a numeric or ordered factor vector with observed `X` values.
`y`	a numeric or ordered factor vector with observed `Y` values, of the same length as `x`.
`variant`	either 'DDP', 'ADP'.
`K`	the size of the KxK partition. Must be between 2 and the square root of the length of the input vectors.
`correct.mi.bias`	a boolean specifying whether to apply the Miller-Madow bias correction that may be useful for mutual information estimation.
`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). Note this only effects the specialized algorithms for small `K` as described below.
`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). Note this only effects the specialized algorithms for small `K` as described below.

Details

This is an implementation of the DDP (data-derived partition) / ADP (all data partitions) class of univariate distribution-free tests (Heller et al., 2014). This class extends Hoeffding's test by considering data-derived (or all) partitions of an arbitrary order `KxK` of the paired sample (or rank) sapce. Being distribution-free, it does not require resampling for computing p-values, making it considerably faster to apply than the HHG test (see example below).

Specialized algorithms are used for DDP with `K=2,3,4`, and ADP with `K=2`, which are of `O(n^1), O(n^2), O(n^3)` and `O(n^2)` time complexity, respectively (with `n = length(x)`). These algorithms compute, in addition to the sum-aggregated statistics which are available for any `K`, statistics based on max aggregation. These algorithms take `w.sum` and `w.max` into account.

For generating p-values, either simulate your own null tables as shown in the example below, or, if your sample size (`length(x)`) is available, use the pre-computed tables linked from R. Heller's homepage at `http://www.math.tau.ac.il/~ruheller/Software.html`. See the README file bundled with the tables for current details on availability and storage format. At the time of writing tables are available for sample sizes 30, 40, 50, 100, and 300. For each sample size, DDP and ADP tests with `K = 2:floor(sqrt(length(x)))` were computed. The tables contain 1e6 replicates (i.e., support p-values down to 1e-6), except for sample size 300 where currently there are 2e4 samples.

Value

The specialized cases return four statistics:

`sum.chisq` - normalized sum of Pearson chi-squared statistics from the KxK contingency tables considered.

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

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

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

Sum statistics are normalized by the total number of partitions multiplied by `length(x)`, which is the scale for measuring mutual information (in natural digits, aka 'nats').

As described in Heller et al. (2014), DDP with `K > 4` and ADP with `K > 2` resort to by-cell rather than by-partition enumeration, and thus cannot currently return 'max' statistics (only 'sum' statistics are returned).

Note

The tests are consistent for continuous random variables, i.e., when there are no ties. Any ties that do exist in `x` and `y` are automatically resolved by random assignment of ranks (see the built-in function `rank`).

Author(s)

Shachar Kaufman.

References

Heller, R., Heller, Y., Kaufman S., & Gorfine, M. (2014). Consistent distribution-free tests of association between univariate random variables. *arXiv:1308.1559*.

Examples

```
# Background: Similar expression (activity) patterns of genes indicate that
# those genes share a common function. It is thus possible to infer the function
# of genes where it is currently unknown, or to learn about pathways involving
# multiple genes working in concert. Hopefully this knowledge can then be used
# to develop better crops, new treatments to disease, and so on.

# In this example we use the DDP test to find pairs of genes with associated
# expression patterns in the data of Hughes et al. (2000). Specifically, we will
# use the 3x3 DDP test variant based on summation of the likelihood ratio.

data(hughes)
nr.genes = nrow(hughes)
nr.samples = ncol(hughes)

# We want to test each pair of genes for association.
nr.tests = choose(nr.genes, 2)

# Estimate the critical test statistic value by Monte-Carlo simulation of the
# null distribution. We use enough null replicates in order to ensure accurate
# estimation.
bonferroni.level = 0.05 / nr.tests
nr.null.replicates = 100 / bonferroni.level

if (0) {
  # This takes a while to compute but only needs to be done once (and is easy to
  # parallelize over many CPUs).
```

```
    ddp.nullsim = rep(Inf, nr.null.replicates)
    x = 1:nr.samples # it is enough to permute the y vector

    for (i in 2:nr.null.replicates) {
      y = sample(nr.samples)
      ddp.nullsim[i] = xdp.test(x, y, variant = 'DDP', K = 3)$sum.lr
    }

    ddp.critical.value = quantile(ddp.nullsim, probs = 1 - bonferroni.level)
  } else {
    # In the interest of saving time, this is what we get
    # NOTE: Big Monte-Carlo simulated null distributions are available (see documentation)
    ddp.critical.value = 0.01569162
  }

  # Now we compute the observed test statistics for all pairs
  compute.xdp.for.pair = function(pair.idxs) {
    x = as.numeric(hughes[pair.idxs[1], ])
    y = as.numeric(hughes[pair.idxs[2], ])
    return (xdp.test(x, y, variant = 'DDP', K = 3)$sum.lr)
  }

  ddp.obs = combn(nr.genes, 2, compute.xdp.for.pair, simplify = TRUE)

  # And identify any significantly associated pairs (at the conservative
  # Bonferroni level maintaining the FWER at 0.05)
  ddp.significant = (ddp.obs > ddp.critical.value)

  summary(ddp.significant)

  # Result: There are 26 pairs out of the 45 performed here, where there is
  # evidence of associated patterns! Had we analyzed this data with Pearson or
  # Spearman correlation based testing, we would find only a small fraction of
  # these associated pairs (see Heller et al., 2014).
```

Index