Package: coin (via r-universe)

May 25, 2026

Version 1.4-4

Date 20YY-MM-DD

Title Conditional Inference Procedures in a Permutation Test Framework

Description Conditional inference procedures for the general independence problem including two-sample, K-sample (non-parametric ANOVA), correlation, censored, ordered and multivariate problems described in <doi:10.18637/jss.v028.i08>.

Depends R (>= 3.6.0), survival

Imports methods, parallel, stats, stats4, utils, libcoin (>= 1.0-9), matrixStats (>= 0.54.0), modeltools (>= 0.2-9), mvtnorm (>= 1.0-5), multcomp

Suggests xtable, e1071, vcd, TH.data (>= 1.0-7), bibtex

LinkingTo libcoin (>= 1.0-9)

LazyData yes

NeedsCompilation yes

ByteCompile yes

Encoding UTF-8

License GPL-2

URL https://codeberg.org/thothorn/coin/

Repository https://thothorn.r-universe.dev

Date/Publication 2026-05-25 13:38:07 UTC

RemoteUrl https://codeberg.org/thothorn/coin

RemoteRef HEAD

RemoteSha 16ce78bd7e2432addd377e05baf17cd7f43c3c1d

RemoteSubdir pkg/coin

Contents

coin-package *General Information on the* **coin** *Package*

Description

The **coin** package provides an implementation of a general framework for conditional inference procedures commonly known as *permutation tests*. The framework was developed by Strasser and Weber (1999) and is based on a multivariate linear statistic and its conditional expectation, covariance and limiting distribution. These results are utilized to construct tests of independence between two sets of variables.

The package does not only provide a flexible implementation of the abstract framework, but also provides a large set of convenience functions implementing well-known as well as lesser-known classical and non-classical test procedures within the framework. Many of the tests presented in prominent text books, such as Hollander and Wolfe (1999) or Agresti (2002), are immediately available or can be implemented without much effort. Examples include linear rank statistics for the two- and K-sample location and scale problem against ordered and unordered alternatives including post-hoc tests for arbitrary contrasts, tests of independence for contingency tables, two- and K-sample tests for censored data, tests of independence between two continuous variables as well as tests of marginal homogeneity and symmetry. Approximations of the exact null distribution via the limiting distribution or conditional Monte Carlo resampling are available for every test procedure, while the exact null distribution is currently available for univariate two-sample problems only.

The salient parts of the Strasser-Weber framework are elucidated by Hothorn, Hornik, van de Wiel, and Zeileis (2006) and a thorough description of the software implementation is given by Hothorn, Hornik, van de Wiel, and Zeileis (2008).

Author(s)

This package is authored by
Torsten Hothorn <Torsten.Hothorn@R-project.org>,
Kurt Hornik <Kurt.Hornik@R-project.org>,
Mark A. van de Wiel <Mark.vdWiel@vumc.nl>,
Henric Winell <Henric.Winell@statistics.uu.se> and
Achim Zeileis <Achim.Zeileis@R-project.org>.

References

Agresti A (2002). *Categorical Data Analysis*, 2nd edition. John Wiley & Sons, Hoboken, New Jersey.

Hollander M, Wolfe DA (1999). *Nonparametric Statistical Inference*, 2nd edition. John Wiley & Sons, New York, U.S.A.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2006). "A Lego System for Conditional Inference." *The American Statistician*, **60**(3), 257–263. doi:10.1198/000313006X118430.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2008). "Implementing a Class of Permutation Tests: The **coin** Package." *Journal of Statistical Software*, **28**(8), 1–23. doi:10.18637/jss.v028.i08.

Strasser H, Weber C (1999). "On the Asymptotic Theory of Permutation Statistics." *Mathematical Methods of Statistics*, **8**(2), 220–250.

Examples

```
## Not run:
## Generate doxygen documentation if you are interested in the internals:
## Download source package into a temporary directory
tmpdir <- tempdir()
tgz <- download.packages("coin", destdir = tmpdir, type = "source")[2]
## Extract contents
untar(tgz, exdir = tmpdir)
## Run doxygen (assuming it is installed)
wd <- setwd(file.path(tmpdir, "coin"))
system("doxygen inst/doxygen.cfg")
setwd(wd)
## Have fun!
browseURL(file.path(tmpdir, "coin", "inst",
                    "documentation", "html", "index.html"))
## End(Not run)
```

---

alpha *Genetic Components of Alcoholism*

Description

Levels of expressed alpha synuclein mRNA in three groups of allele lengths of NACP-REP1.

Usage

```
alpha
```

Format

A data frame with 97 observations on 2 variables.

`alength` allele length, a factor with levels `"short"`, `"intermediate"` and `"long"`.

`elevel` expression levels of alpha synuclein mRNA.

Details

Various studies have linked alcohol dependence phenotypes to chromosome 4. One candidate gene is NACP (non-amyloid component of plaques), coding for alpha synuclein. Bönsch, Lederer, Reulbach, Hothorn, Kornhuber, and Bleich (2005) found longer alleles of NACP-REP1 in alcohol-dependent patients compared with healthy controls and reported that the allele lengths show some association with levels of expressed alpha synuclein mRNA.

Source

Bönsch D, Lederer T, Reulbach U, Hothorn T, Kornhuber J, Bleich S (2005). "Joint Analysis of the NACP-REP1 Marker Within the Alpha Synuclein Gene Concludes Association with Alcohol Dependence." *Human Molecular Genetics*, **14**(7), 967–971. doi:10.1093/hmg/ddi090.

Examples

```
## Boxplots
boxplot(elevel ~ alength, data = alpha)

## Asymptotic Kruskal-Wallis test
kruskal_test(elevel ~ alength, data = alpha)

## Asymptotic Kruskal-Wallis test using midpoint scores
kruskal_test(elevel ~ alength, data = alpha,
             scores = list(alength = c(2, 7, 11)))

## Asymptotic score-independent test
## Winell and Lindbaeck (2018)
(it <- independence_test(elevel ~ alength, data = alpha,
                         ytrafo = function(data)
                             trafo(data, numeric_trafo = rank_trafo),
                         xtrafo = function(data)
                             trafo(data, factor_trafo = function(x)
                                 zheng_trafo(as.ordered(x)))))

## Extract the "best" set of scores
ss <- statistic(it, type = "standardized")
idx <- which(abs(ss) == max(abs(ss)), arr.ind = TRUE)
ss[idx[1], idx[2], drop = FALSE]
```

---

alzheimer *Smoking and Alzheimer's Disease*

Description

A case-control study of smoking and Alzheimer's disease.

Usage

```
alzheimer
```

Format

A data frame with 538 observations on 3 variables.

`smoking` a factor with levels `"None"`, `"<10"`, `"10-20"` and `">20"` (cigarettes per day).

`disease` a factor with levels `"Alzheimer"`, `"Other dementias"` and `"Other diagnoses"`.

`gender` a factor with levels `"Female"` and `"Male"`.

Details

Subjects with Alzheimer's disease are compared to two different control groups with respect to smoking history. The data are given in Salib and Hillier (1997, Tab. 4).

Source

Salib E, Hillier V (1997). "A Case-Control Study of Smoking and Alzheimer's Disease." *International Journal of Geriatric Psychiatry*, **12**, 295–300.

Examples

```
## Spineplots
op <- par(no.readonly = TRUE) # save current settings
layout(matrix(1:2, ncol = 2))
spineplot(disease ~ smoking, data = alzheimer,
          subset = gender == "Male", main = "Male")
spineplot(disease ~ smoking, data = alzheimer,
          subset = gender == "Female", main = "Female")
par(op) # reset

## Asymptotic Cochran-Mantel-Haenszel test
cmh_test(disease ~ smoking | gender, data = alzheimer)
```

---

asat *Toxicological Study on Female Wistar Rats*

Description

Measurements of the liver enzyme aspartate aminotransferase (ASAT) for a new compound and a control group of 34 female Wistar rats.

Usage

```
asat
```

Format

A data frame with 34 observations on 2 variables.

`asat` ASAT values.

`group` a factor with levels `"Compound"` and `"Control"`.

Details

The aim of this toxicological study is the proof of safety for the new compound. The data were originally given in Hothorn (1992) and later reproduced by Hauschke, Kieser, and Hothorn (1999), see also Pflüger and Hothorn (2002).

References

Hauschke D, Kieser M, Hothorn LA (1999). "Proof of Safety in Toxicology Based on the Ratio of Two Means for Normally Distributed Data." *Biometrical Journal*, **41**(3), 295–304. ISSN 1521-4036. doi:10.1002/(SICI)15214036(199906)41:3<295::AIDBIMJ295>3.0.CO;22.

Hothorn LA (1992). "Biometrische Analyse Toxikologischer Untersuchungen." In Adam J (ed.), *Statistisches Know-How in der Medizinischen Forschung*, 475–590.

Pflüger R, Hothorn T (2002). "Assessing Equivalence Tests with Respect to their Expected p-Value." *Biometrical Journal*, **44**(8), 1015–1027. doi:10.1002/bimj.200290001.

Examples

```
## Proof-of-safety based on ratio of medians (Pflueger and Hothorn, 2002)
## One-sided exact Wilcoxon-Mann-Whitney test
wt <- wilcox_test(I(log(asat)) ~ group, data = asat,
                  distribution = "exact", alternative = "less",
                  conf.int = TRUE)

## One-sided confidence set
## Note: Safety cannot be concluded since the effect of the compound
##       exceeds 20 % of the control median
exp(confint(wt)$conf.int)
```

`ContingencyTests` *Tests of Independence in Two- or Three-Way Contingency Tables*

Description

Testing the independence of two nominal or ordered factors.

Usage

```
## S3 method for class 'formula'
chisq_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
chisq_test(object, ...)
## S3 method for class 'IndependenceProblem'
chisq_test(object, ...)

## S3 method for class 'formula'
cmh_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
cmh_test(object, ...)
## S3 method for class 'IndependenceProblem'
cmh_test(object, ...)

## S3 method for class 'formula'
lbl_test(formula, data, subset = NULL, weights = NULL, ...)
```

```
## S3 method for class 'table'
lbl_test(object, ...)
## S3 method for class 'IndependenceProblem'
lbl_test(object, ...)
```

Arguments

formula	a formula of the form y ~ x | block where y and x are factors and block is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations.
object	an object inheriting from classes "table" or "IndependenceProblem".
...	further arguments to be passed to independence_test().

Details

chisq_test(), cmh_test() and lbl_test() provide the Pearson chi-squared test, the generalized Cochran-Mantel-Haenszel test and the linear-by-linear association test. A general description of these methods is given by Agresti (2002).

The null hypothesis of independence, or conditional independence given block, between y and x is tested.

If y and/or x are ordered factors, the default scores, 1:nlevels(y) and 1:nlevels(x), respectively, can be altered using the scores argument (see independence_test()); this argument can also be used to coerce nominal factors to class "ordered". (lbl_test() coerces to class "ordered" under any circumstances.) If both y and x are ordered factors, a linear-by-linear association test is computed and the direction of the alternative hypothesis can be specified using the alternative argument. For the Pearson chi-squared test, this extension was given by Yates (1948) who also discussed the situation when either the response or the covariate is an ordered factor; see also Cochran (1954) and Armitage (1955) for the particular case when y is a binary factor and x is ordered. The Mantel-Haenszel statistic (Mantel and Haenszel 1959) was similarly extended by Mantel (1963) and Landis, Heyman, and Koch (1978).

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (distribution = "asymptotic"). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for univariate two-sample problems by setting distribution to "approximate" or "exact", respectively. See asymptotic(), approximate() and exact() for details.

Value

An object inheriting from class "IndependenceTest".

Note

The exact versions of the Pearson chi-squared test and the generalized Cochran-Mantel-Haenszel test do not necessarily result in the same p-value as Fisher's exact test (Davis 1986).

References

Agresti A (2002). *Categorical Data Analysis*, 2nd edition. John Wiley & Sons, Hoboken, New Jersey.

Armitage P (1955). "Tests for Linear Trends in Proportions and Frequencies." *Biometrics*, **11**(3), 375–386. doi:10.2307/3001775.

Cochran WG (1954). "Some Methods for Strengthening the Common χ^2 Tests." *Biometrics*, **10**(4), 417–451. doi:10.2307/3001616.

Davis LJ (1986). "Exact Tests for 2×2 Contingency Tables." *The American Statistician*, **40**(2), 139–141. doi:10.1080/00031305.1986.10475377.

Landis JR, Heyman ER, Koch GG (1978). "Average Partial Association in Three-Way Contingency Tables: A Review and Discussion of Alternative Tests." *International Statistical Review / Revue Internationale de Statistique*, **46**(3), 237–254. doi:10.2307/1402373.

Mantel N (1963). "Chi-Square Tests with One Degree of Freedom; Extensions of the Mantel-Haenszel Procedure." *Journal of the American Statistical Association*, **58**(303), 690–700. doi:10.1080/01621459.1963.10500879.

Mantel N, Haenszel W (1959). "Statistical Aspects of the Analysis of Data From Retrospective Studies of Disease." *JNCI: Journal of the National Cancer Institute*, **22**(4), 719–748. doi:10.1093/jnci/22.4.719.

Yates F (1948). "The Analysis of Contingency Tables with Groupings Based on Quantitative Characters." *Biometrika*, **35**(1–2), 176–181. doi:10.1093/biomet/35.12.176.

Examples

```
## Example data
## Davis (1986, p. 140)
davis <- matrix(
    c(3,  6,
      2, 19),
    nrow = 2, byrow = TRUE
)
davis <- as.table(davis)

## Asymptotic Pearson chi-squared test
chisq_test(davis)
chisq.test(davis, correct = FALSE) # same as above

## Approximative (Monte Carlo) Pearson chi-squared test
ct <- chisq_test(davis,
                 distribution = approximate(nresample = 10000))
pvalue(ct)             # standard p-value
midpvalue(ct)          # mid-p-value
pvalue_interval(ct)    # p-value interval
size(ct, alpha = 0.05) # test size at alpha = 0.05 using the p-value

## Exact Pearson chi-squared test (Davis, 1986)
## Note: disagrees with Fisher's exact test
ct <- chisq_test(davis,
                 distribution = "exact")
```

```
pvalue(ct)             # standard p-value
midpvalue(ct)          # mid-p-value
pvalue_interval(ct)    # p-value interval
size(ct, alpha = 0.05) # test size at alpha = 0.05 using the p-value
fisher.test(davis)


## Laryngeal cancer data
## Agresti (2002, p. 107, Tab. 3.13)
cancer <- matrix(
    c(21, 2,
      15, 3),
    nrow = 2, byrow = TRUE,
    dimnames = list(
        "Treatment" = c("Surgery", "Radiation"),
           "Cancer" = c("Controlled", "Not Controlled")
    )
)
cancer <- as.table(cancer)

## Exact Pearson chi-squared test (Agresti, 2002, p. 108, Tab. 3.14)
## Note: agrees with Fishers's exact test
(ct <- chisq_test(cancer,
                  distribution = "exact"))
midpvalue(ct)          # mid-p-value
pvalue_interval(ct)    # p-value interval
size(ct, alpha = 0.05) # test size at alpha = 0.05 using the p-value
fisher.test(cancer)


## Homework conditions and teacher's rating
## Yates (1948, Tab. 1)
yates <- matrix(
    c(141, 67, 114, 79, 39,
      131, 66, 143, 72, 35,
       36, 14,  38, 28, 16),
    byrow = TRUE, ncol = 5,
    dimnames = list(
           "Rating" = c("A", "B", "C"),
        "Condition" = c("A", "B", "C", "D", "E")
    )
)
yates <- as.table(yates)

## Asymptotic Pearson chi-squared test (Yates, 1948, p. 176)
chisq_test(yates)

## Asymptotic Pearson-Yates chi-squared test (Yates, 1948, pp. 180-181)
## Note: 'Rating' and 'Condition' as ordinal
(ct <- chisq_test(yates,
                  alternative = "less",
                  scores = list("Rating" = c(-1, 0, 1),
                                "Condition" = c(2, 1, 0, -1, -2))))
```

```
statistic(ct)^2 # chi^2 = 2.332

## Asymptotic Pearson-Yates chi-squared test (Yates, 1948, p. 181)
## Note: 'Rating' as ordinal
chisq_test(yates,
           scores = list("Rating" = c(-1, 0, 1))) # Q = 3.825


## Change in clinical condition and degree of infiltration
## Cochran (1954, Tab. 6)
cochran <- matrix(
    c(11,  7,
      27, 15,
      42, 16,
      53, 13,
      11,  1),
    byrow = TRUE, ncol = 2,
    dimnames = list(
              "Change" = c("Marked", "Moderate", "Slight",
                           "Stationary", "Worse"),
        "Infiltration" = c("0-7", "8-15")
    )
)
cochran <- as.table(cochran)

## Asymptotic Pearson chi-squared test (Cochran, 1954, p. 435)
chisq_test(cochran) # X^2 = 6.88

## Asymptotic Cochran-Armitage test (Cochran, 1954, p. 436)
## Note: 'Change' as ordinal
(ct <- chisq_test(cochran,
                  scores = list("Change" = c(3, 2, 1, 0, -1))))
statistic(ct)^2 # X^2 = 6.66


## Change in size of ulcer crater for two treatment groups
## Armitage (1955, Tab. 2)
armitage <- matrix(
    c( 6, 4, 10, 12,
      11, 8,  8,  5),
    byrow = TRUE, ncol = 4,
    dimnames = list(
        "Treatment" = c("A", "B"),
           "Crater" = c("Larger", "< 2/3 healed",
                        ">= 2/3 healed", "Healed")
    )
)
armitage <- as.table(armitage)

## Approximative (Monte Carlo) Pearson chi-squared test (Armitage, 1955, p. 379)
chisq_test(armitage,
           distribution = approximate(nresample = 10000)) # chi^2 = 5.91
```

```
## Approximative (Monte Carlo) Cochran-Armitage test (Armitage, 1955, p. 379)
(ct <- chisq_test(armitage,
                  distribution = approximate(nresample = 10000),
                  scores = list("Crater" = c(-1.5, -0.5, 0.5, 1.5))))
statistic(ct)^2 # chi_0^2 = 5.26


## Relationship between job satisfaction and income stratified by gender
## Agresti (2002, p. 288, Tab. 7.8)

## Asymptotic generalized Cochran-Mantel-Haenszel test (Agresti, p. 297)
(ct <- cmh_test(jobsatisfaction)) # CMH = 10.2001

## The standardized linear statistic
statistic(ct, type = "standardized")

## The standardized linear statistic for each block
statistic(ct, type = "standardized", partial = TRUE)

## Asymptotic generalized Cochran-Mantel-Haenszel test (Agresti, p. 297)
## Note: 'Job.Satisfaction' as ordinal
cmh_test(jobsatisfaction,
         scores = list("Job.Satisfaction" = c(1, 3, 4, 5))) # L^2 = 9.0342

## Asymptotic linear-by-linear association test (Agresti, p. 297)
## Note: 'Job.Satisfaction' and 'Income' as ordinal
(lt <- lbl_test(jobsatisfaction,
                scores = list("Job.Satisfaction" = c(1, 3, 4, 5),
                              "Income" = c(3, 10, 20, 35))))
statistic(lt)^2 # M^2 = 6.1563

## The standardized linear statistic
statistic(lt, type = "standardized")

## The standardized linear statistic for each block
statistic(lt, type = "standardized", partial = TRUE)
```

CorrelationTests *Correlation Tests*

Description

Testing the independence of two numeric variables.

Usage

```
## S3 method for class 'formula'
spearman_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
spearman_test(object, distribution = c("asymptotic", "approximate", "none"), ...)
```

```
## S3 method for class 'formula'
fisyat_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
fisyat_test(object, distribution = c("asymptotic", "approximate", "none"),
            ties.method = c("mid-ranks", "average-scores"), ...)

## S3 method for class 'formula'
quadrant_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
quadrant_test(object, distribution = c("asymptotic", "approximate", "none"),
              mid.score = c("0", "0.5", "1"), ...)

## S3 method for class 'formula'
koziol_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
koziol_test(object, distribution = c("asymptotic", "approximate", "none"),
            ties.method = c("mid-ranks", "average-scores"), ...)
```

Arguments

formula	a formula of the form y ~ x | block where y and x are numeric variables and block is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations.
object	an object inheriting from class "IndependenceProblem".
distribution	a character, the conditional null distribution of the test statistic can be approximated by its asymptotic distribution ("asymptotic", default) or via Monte Carlo resampling ("approximate"). Alternatively, the functions asymptotic or approximate can be used. Computation of the null distribution can be suppressed by specifying "none".
ties.method	a character, the method used to handle ties: the score generating function either uses mid-ranks ("mid-ranks", default) or averages the scores of randomly broken ties ("average-scores").
mid.score	a character, the score assigned to observations exactly equal to the median: either 0 ("0", default), 0.5 ("0.5") or 1 ("1"); see median_test().
...	further arguments to be passed to independence_test().

Details

spearman_test(), fisyat_test(), quadrant_test() and koziol_test() provide the Spearman correlation test, the Fisher-Yates correlation test using van der Waerden scores, the quadrant test and the Koziol-Nemec test. A general description of these methods is given by Hájek, Šidák,

and Sen (1999, Sec. 4.6). The Koziol-Nemec test was suggested by Koziol and Nemec (1979). For the adjustment of scores for tied values see Hájek et al. (1999, pp. 133–135).

The null hypothesis of independence, or conditional independence given `block`, between `y` and `x` is tested.

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (`distribution = "asymptotic"`). Alternatively, the distribution can be approximated via Monte Carlo resampling by setting `distribution` to `"approximate"`. See `asymptotic()` and `approximate()` for details.

Value

An object inheriting from class `"IndependenceTest"`.

References

Hájek J, Šidák Z, Sen PK (1999). *Theory of Rank Tests*, 2nd edition. Academic Press, London, UK.

Koziol JA, Nemec AF (1979). "On a Cramér-von Mises Type Statistic for Testing Bivariate Independence." *Canadian Journal of Statistics*, **7**(1), 43–52. doi:10.2307/3315014.

Examples

```
## Asymptotic Spearman test
spearman_test(CONT ~ INTG, data = USJudgeRatings)

## Asymptotic Fisher-Yates test
fisyat_test(CONT ~ INTG, data = USJudgeRatings)

## Asymptotic quadrant test
quadrant_test(CONT ~ INTG, data = USJudgeRatings)

## Asymptotic Koziol-Nemec test
koziol_test(CONT ~ INTG, data = USJudgeRatings)
```

CWD *Coarse Woody Debris*

Description

Carbon flux on six pieces of wood.

Usage

```
CWD
```

Format

A data frame with 13 observations on 8 variables.

`sample2` carbon flux measurement for 2nd piece of wood.

`sample3` carbon flux measurement for 3rd piece of wood.

`sample4` carbon flux measurement for 4th piece of wood.

`sample6` carbon flux measurement for 6th piece of wood.

`sample7` carbon flux measurement for 7th piece of wood.

`sample8` carbon flux measurement for 8th piece of wood.

`trend` measurement day (in days from beginning).

`time` date of measurement.

Details

Coarse woody debris (CWD, dead wood greater than 10 cm in diameter) is a large stock of carbon in tropical forests, yet the flux of carbon out of this pool, via respiration, is poorly resolved (Chambers, Schimel, and Nobre 2001). The heterotrophic process involved in CWD respiration should respond to reductions in moisture availability, which occurs during dry season (Chambers et al. 2001).

CWD respiration measurements were taken in a tropical forest in west French Guiana, which experiences extreme contrasts in wet and dry season (Bonal, Bosc, Ponton, Goret, Burban, Gross, Bonnefond, Elbers, Longdoz, Epron, Guehl, and Granier 2008). An infrared gas analyzer and a clear chamber sealed to the wood surface were used to measure the flux of carbon out of the wood (Stahl, Burban, Goret, and Bonal 2011). Measurements were repeated 13 times, from July to November 2011, on six pieces of wood during the transition into and out of the dry season. The aim is to assess if there were shifts in the CWD respiration of any of the pieces in response to the transition into (early August) and out of (late October) the dry season.

Zeileis and Hothorn (2013) investigated the six-variate series of CO_2 reflux, aiming to find out whether the reflux had changed over the sampling period in at least one of the six wood pieces.

Source

The coarse woody debris respiration data were kindly provided by Lucy Rowland (School of GeoSciences, University of Edinburgh).

References

Bonal D, Bosc A, Ponton S, Goret J, Burban B, Gross P, Bonnefond J, Elbers J, Longdoz B, Epron D, Guehl J, Granier A (2008). "Impact of Severe Dry Season on Net Ecosystem Exchange in the Neotropical Rainforest of French Guiana." *Global Change Biology*, **14**(8), 1917–1933. doi:10.1111/j.13652486.2008.01610.x.

Chambers JQ, Schimel JP, Nobre AD (2001). "Respiration from Coarse Wood Litter in Central Amazon Forests." *Biogeochemistry*, **52**(2), 115–131. doi:10.1023/a:1006473530673.

Stahl C, Burban B, Goret J, Bonal D (2011). "Seasonal Variations in Stem CO2 Efflux in the Neotropical Rainforest of French Guiana." *Annals of Forest Science*, **68**(4), 771–782. doi:10.1007/s1359501100742.

Zeileis A, Hothorn T (2013). "A Toolbox of Permutation Tests for Structural Change." *Statistical Papers*, **54**(4), 931–954. doi:10.1007/s0036201305034.

Examples

```
## Zeileis and Hothorn (2013, pp. 942-944)
## Approximative (Monte Carlo) maximally selected statistics
CWD[1:6] <- 100 * CWD[1:6] # scaling (to avoid harmless warning)
mt <- maxstat_test(sample2 + sample3 + sample4 +
                   sample6 + sample7 + sample8 ~ trend, data = CWD,
                   distribution = approximate(nresample = 100000))

## Absolute maximum of standardized statistics (t = 3.08)
statistic(mt)

## 5% critical value (t_0.05 = 2.86)
(c <- qperm(mt, 0.95))

## Only 'sample8' exceeds the 5% critical value
sts <- statistic(mt, type = "standardized")
idx <- which(sts > c, arr.ind = TRUE)
sts[unique(idx[, 1]), unique(idx[, 2]), drop = FALSE]
```

---

expectation-methods *Extraction of the Expectation, Variance and Covariance of the Linear Statistic*

Description

Methods for extraction of the expectation, variance and covariance of the linear statistic.

Usage

```
## S4 method for signature 'IndependenceLinearStatistic'
expectation(object, partial = FALSE, ...)
## S4 method for signature 'IndependenceTest'
expectation(object, partial = FALSE, ...)

## S4 method for signature 'Variance'
variance(object, ...)
## S4 method for signature 'CovarianceMatrix'
variance(object, ...)
## S4 method for signature 'IndependenceLinearStatistic'
variance(object, partial = FALSE, ...)
## S4 method for signature 'IndependenceTest'
variance(object, partial = FALSE, ...)

## S4 method for signature 'CovarianceMatrix'
covariance(object, ...)
## S4 method for signature 'IndependenceLinearStatistic'
covariance(object, invert = FALSE, partial = FALSE, ...)
## S4 method for signature 'QuadTypeIndependenceTestStatistic'
```

```
covariance(object, invert = FALSE, partial = FALSE, ...)
## S4 method for signature 'IndependenceTest'
covariance(object, invert = FALSE, partial = FALSE, ...)
```

Arguments

object	an object from which the expectation, variance or covariance of the linear statistic can be extracted.
partial	a logical indicating that the partial result for each block should be extracted. Defaults to FALSE.
invert	a logical indicating that the Moore-Penrose inverse of the covariance should be extracted. Defaults to FALSE.
...	further arguments (currently ignored).

Details

The methods expectation, variance and covariance extract the expectation, variance and covariance, respectively, of the linear statistic.

For tests of conditional independence within blocks, the partial result for each block is obtained by setting partial = TRUE.

Value

The expectation, variance or covariance of the linear statistic extracted from object. A matrix or array.

Examples

```
## Example data
dta <- data.frame(
    y = gl(3, 2),
    x = sample(gl(3, 2))
)

## Asymptotic Cochran-Mantel-Haenszel Test
ct <- cmh_test(y ~ x, data = dta)

## The linear statistic, i.e., the contingency table...
(T <- statistic(ct, type = "linear"))

## ...and its expectation...
(mu <- expectation(ct))

## ...and variance...
(sigma <- variance(ct))

## ...and covariance...
(Sigma <- covariance(ct))

## ...and its inverse
```

```
(SigmaPlus <- covariance(ct, invert = TRUE))

## The standardized contingency table...
(T - mu) / sqrt(sigma)

## ...is identical to the standardized linear statistic
statistic(ct, type = "standardized")

## The quadratic form...
U <- as.vector(T - mu)
U %*% SigmaPlus %*% U

## ...is identical to the test statistic
statistic(ct, type = "test")
```

`glioma` *Malignant Glioma Pilot Study*

Description

A non-randomized pilot study on malignant glioma patients with pretargeted adjuvant radioimmunotherapy using yttrium-90-biotin.

Usage

```
glioma
```

Format

A data frame with 37 observations on 7 variables.

`no.` patient number.

`age` patient age (years).

`sex` a factor with levels `"F"` (Female) and `"M"` (Male).

`histology` a factor with levels `"GBM"` (grade IV) and `"Grade3"` (grade III).

`group` a factor with levels `"Control"` and `"RIT"`.

`event` status indicator for `time`: `FALSE` for right-censored observations and `TRUE` otherwise.

`time` survival time (months).

Details

The primary endpoint of this small pilot study is survival. Since the survival times are tied, the classical asymptotic logrank test may be inadequate in this setup. Therefore, a permutation test using Monte Carlo resampling was computed in the original paper. The data are taken from Tables 1 and 2 of Grana, Chinol, Robertson, Mazzetta, Bartolomei, De Cicco, Fiorenza, Gatti, Caliceti, and Paganelli (2002).

Source

Grana C, Chinol M, Robertson C, Mazzetta C, Bartolomei M, De Cicco C, Fiorenza M, Gatti M, Caliceti P, Paganelli G (2002). "Pretargeted Adjuvant Radioimmunotherapy with Yttrium-90-biotin in Malignant Glioma Patients: A Pilot Study." *British Journal of Cancer*, **86**(2), 207–212. doi:10.1038/sj.bjc.6600047.

Examples

```
## Grade III glioma
g3 <- subset(glioma, histology == "Grade3")

## Plot Kaplan-Meier estimates
op <- par(no.readonly = TRUE) # save current settings
layout(matrix(1:2, ncol = 2))
plot(survfit(Surv(time, event) ~ group, data = g3),
     main = "Grade III Glioma", lty = 2:1,
     ylab = "Probability", xlab = "Survival Time in Month",
     xlim = c(-2, 72))
legend("bottomleft", lty = 2:1, c("Control", "Treated"), bty = "n")

## Exact logrank test
logrank_test(Surv(time, event) ~ group, data = g3,
             distribution = "exact")


## Grade IV glioma
gbm <- subset(glioma, histology == "GBM")

## Plot Kaplan-Meier estimates
plot(survfit(Surv(time, event) ~ group, data = gbm),
     main = "Grade IV Glioma", lty = 2:1,
     ylab = "Probability", xlab = "Survival Time in Month",
     xlim = c(-2, 72))
legend("topright", lty = 2:1, c("Control", "Treated"), bty = "n")
par(op) # reset

## Exact logrank test
logrank_test(Surv(time, event) ~ group, data = gbm,
             distribution = "exact")


## Stratified approximative (Monte Carlo) logrank test
logrank_test(Surv(time, event) ~ group | histology, data = glioma,
             distribution = approximate(nresample = 10000))
```

Description

A randomized clinical trial in gastric cancer.

Usage

```
GTSG
```

Format

A data frame with 90 observations on 3 variables.

`time` survival time (days).

`event` status indicator for `time`: `0` for right-censored observations and `1` otherwise.

`group` a factor with levels `"Chemotherapy+Radiation"` and `"Chemotherapy"`.

Details

A clinical trial comparing chemotherapy alone versus a combination of chemotherapy and radiation therapy in the treatment of locally advanced, nonresectable gastric carcinoma (Stablein, Carter, and Novak 1981).

Results of the procedures by Moreau, Maccario, Lellouch, and Huber (1992) and Shen and Le (2000) are presented in the example section.

Note

There is substantial separation between the estimated survival distributions at 8 to 10 months, but by month 26 the distributions intersect.

References

Moreau T, Maccario J, Lellouch J, Huber C (1992). "Weighted Log Rank Statistics for Comparing Two Distributions." *Biometrika*, **79**(1), 195–198. doi:10.1093/biomet/79.1.195.

Shen W, Le CT (2000). "Linear Rank Tests for Censored Survival Data." *Communications in Statistics - Simulation and Computation*, **29**(1), 21–36. doi:10.1080/03610910008813599.

Stablein DM, Carter WH, Novak JW (1981). "Analysis of Survival Data with Nonproportional Hazard Functions." *Controlled Clinical Trials*, **2**(2), 149–159. doi:10.1016/01972456(81)900052.

Examples

```
## Plot Kaplan-Meier estimates
plot(survfit(Surv(time / (365.25 / 12), event) ~ group, data = GTSG),
     lty = 1:2, ylab = "% Survival", xlab = "Survival Time in Months")
legend("topright", lty = 1:2,
       c("Chemotherapy+Radiation", "Chemotherapy"), bty = "n")

## Asymptotic logrank test
logrank_test(Surv(time, event) ~ group, data = GTSG)

## Asymptotic Prentice test
```

```
logrank_test(Surv(time, event) ~ group, data = GTSG, type = "Prentice")

## Asymptotic test against Weibull-type alternatives (Moreau et al., 1992)
moreau_weight <- function(time, n.risk, n.event)
    1 + log(-log(cumprod(n.risk / (n.risk + n.event))))

independence_test(Surv(time, event) ~ group, data = GTSG,
                  ytrafo = function(data)
                      trafo(data, surv_trafo = function(y)
                          logrank_trafo(y, weight = moreau_weight)))

## Asymptotic test against crossing-curve alternatives (Shen and Le, 2000)
shen_trafo <- function(x)
    ansari_trafo(logrank_trafo(x, type = "Prentice"))

independence_test(Surv(time, event) ~ group, data = GTSG,
                  ytrafo = function(data)
                      trafo(data, surv_trafo = shen_trafo))
```

`hohnloser` *Left Ventricular Ejection Fraction*

Description

Left ventricular ejection fraction in patients with malignant ventricular tachyarrhythmias including recurrence-free month and censoring.

Usage

```
hohnloser
```

Format

A data frame with 94 observations on 3 variables.

`EF` ejection fraction (%).

`time` recurrence-free month.

`event` status indicator for `time`: `0` for right-censored observations and `1` otherwise.

Details

The data published by Hohnloser, Raeder, Podrid, Graboys, and Lown (1987) were used by Lausen and Schumacher (1992) to illustrate the use of maximally selected statistics.

References

Hohnloser SH, Raeder EA, Podrid PJ, Graboys TB, Lown B (1987). "Predictors of Antiarrhythmic Drug Efficacy in Patients with Malignant Ventricular Tachyarrhythmias." *American Heart Journal*, **114**(1), 1–7. doi:10.1016/00028703(87)902997.

Lausen B, Schumacher M (1992). "Maximally Selected Rank Statistics." *Biometrics*, **48**(1), 73–85. doi:10.2307/2532740.

Examples

```
## Asymptotic maximally selected logrank statistics
maxstat_test(Surv(time, event) ~ EF, data = hohnloser)
```

IndependenceLinearStatistic-class

Class "IndependenceLinearStatistic"

Description

Objects of class "IndependenceLinearStatistic" represent the linear statistic and the transformed and original data structures corresponding to an independence problem.

Objects from the Class

Objects can be created by calls of the form

```
new("IndependenceLinearStatistic", object, ...)
```

where `object` is an object of class "IndependenceTestProblem".

Slots

`linearstatistic`: Object of class "matrix". The linear statistic for each block.

`expectation`: Object of class "matrix". The expectation of the linear statistic for each block.

`covariance`: Object of class "matrix". The lower triangular elements of the covariance of the linear statistic for each block.

`xtrans`: Object of class "matrix". The transformed `x`.

`ytrans`: Object of class "matrix". The transformed `y`.

`xtrafo`: Object of class "function". The regression function for `x`.

`ytrafo`: Object of class "function". The influence function for `y`.

`x`: Object of class "data.frame". The variables `x`.

`y`: Object of class "data.frame". The variables `y`.

`block`: Object of class "factor". The block structure.

`weights`: Object of class "numeric". The case weights.

Extends

Class "IndependenceTestProblem", directly.
Class "IndependenceProblem", by class "IndependenceTestProblem", distance 2.

Known Subclasses

Class "IndependenceTestStatistic", directly.
Class "MaxTypeIndependenceTestStatistic", by class "IndependenceTestStatistic", distance 2.
Class "QuadTypeIndependenceTestStatistic", by class "IndependenceTestStatistic", distance 2.
Class "ScalarIndependenceTestStatistic", by class "IndependenceTestStatistic", distance 2.

Methods

covariance `signature(object = "IndependenceLinearStatistic")`: See the documentation for `covariance()` for details.

expectation `signature(object = "IndependenceLinearStatistic")`: See the documentation for `expectation()` for details.

initialize `signature(.Object = "IndependenceLinearStatistic")`: See the documentation for `initialize()` (in package **methods**) for details.

statistic `signature(object = "IndependenceLinearStatistic")`: See the documentation for `statistic()` for details.

variance `signature(object = "IndependenceLinearStatistic")`: See the documentation for `variance()` for details.

IndependenceProblem-class

Class "IndependenceProblem"

Description

Objects of class "IndependenceProblem" represent the data structure corresponding to an independence problem.

Objects from the Class

Objects can be created by calls of the form

```
new("IndependenceProblem", x, y, block = NULL, weights = NULL, ...)
```

where `x` and `y` are data frames containing the variables $\mathbf{X}$ and $\mathbf{Y}$, respectively, `block` is an optional factor representing the block structure b and `weights` is an optional integer vector corresponding to the case weights w.

Slots

`x`: Object of class `"data.frame"`. The variables `x`.

`y`: Object of class `"data.frame"`. The variables `y`.

`block`: Object of class `"factor"`. The block structure.

`weights`: Object of class `"numeric"`. The case weights.

Known Subclasses

Class `"IndependenceTestProblem"`, directly.
Class `"SymmetryProblem"`, directly.
Class `"IndependenceLinearStatistic"`, by class `"IndependenceTestProblem"`, distance 2.
Class `"IndependenceTestStatistic"`, by class `"IndependenceTestProblem"`, distance 3.
Class `"MaxTypeIndependenceTestStatistic"`, by class `"IndependenceTestProblem"`, distance 4.
Class `"QuadTypeIndependenceTestStatistic"`, by class `"IndependenceTestProblem"`, distance 4.
Class `"ScalarIndependenceTestStatistic"`, by class `"IndependenceTestProblem"`, distance 4.

Methods

initialize `signature(.Object = "IndependenceProblem")`: See the documentation for `initialize()` (in package **methods**) for details.

IndependenceTest *General Independence Test*

Description

Testing the independence of two sets of variables measured on arbitrary scales.

Usage

```
## S3 method for class 'formula'
independence_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
independence_test(object, ...)
## S3 method for class 'IndependenceProblem'
independence_test(object, teststat = c("maximum", "quadratic", "scalar"),
                  distribution = c("asymptotic", "approximate",
                                   "exact", "none"),
                  alternative = c("two.sided", "less", "greater"),
                  xtrafo = trafo, ytrafo = trafo, scores = NULL,
                  check = NULL, ...)
```

Arguments

formula	a formula of the form y1 + ... + yq ~ x1 + ... + xp | block where y1, ..., yq and x1, ..., xp are measured on arbitrary scales (nominal, ordinal or continuous with or without censoring) and block is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations.
object	an object inheriting from classes "table" or "IndependenceProblem".
teststat	a character, the type of test statistic to be applied: either a maximum statistic ("maximum", default), a quadratic form ("quadratic") or a standardized scalar test statistic ("scalar").
distribution	a character, the conditional null distribution of the test statistic can be approximated by its asymptotic distribution ("asymptotic", default) or via Monte Carlo resampling ("approximate"). Alternatively, the functions asymptotic or approximate can be used. For univariate two-sample problems, "exact" or use of the function exact computes the exact distribution. Computation of the null distribution can be suppressed by specifying "none". It is also possible to specify a function with one argument (an object inheriting from "IndependenceTestStatistic") that returns an object of class "NullDistribution".
alternative	a character, the alternative hypothesis: either "two.sided" (default), "greater" or "less".
xtrafo	a function of transformations to be applied to the variables x1, ..., xp supplied in formula; see 'Details'. Defaults to trafo().
ytrafo	a function of transformations to be applied to the variables y1, ..., yq supplied in formula; see 'Details'. Defaults to trafo().
scores	a named list of scores to be attached to ordered factors; see 'Details'. Defaults to NULL, implying equally spaced scores.
check	a function to be applied to objects of class "IndependenceTest" in order to check for specific properties of the data. Defaults to NULL.
...	further arguments to be passed to or from other methods (currently ignored).

Details

independence_test() provides a general independence test for two sets of variables measured on arbitrary scales. This function is based on the general framework for conditional inference procedures proposed by Strasser and Weber (1999). The salient parts of the Strasser-Weber framework are elucidated by Hothorn, Hornik, van de Wiel, and Zeileis (2006) and a thorough description of the software implementation is given by Hothorn, Hornik, van de Wiel, and Zeileis (2008).

The null hypothesis of independence, or conditional independence given block, between y1, ..., yq and x1, ..., xp is tested.

A vector of case weights, e.g., observation counts, can be supplied through the weights argument and the type of test statistic is specified by the teststat argument. Influence and regression functions, i.e., transformations of y1, ..., yq and x1, ..., xp, are specified by the ytrafo and xtrafo

arguments, respectively; see `trafo()` for the collection of transformation functions currently available. This allows for implementation of both novel and familiar test statistics, e.g., the Pearson χ^2 test, the generalized Cochran-Mantel-Haenszel test, the Spearman correlation test, the Fisher-Pitman permutation test, the Wilcoxon-Mann-Whitney test, the Kruskal-Wallis test and the family of weighted logrank tests for censored data. Furthermore, multivariate extensions such as the multivariate Kruskal-Wallis test (Puri and Sen 1966; Puri and Sen 1971) can be implemented without much effort (see 'Examples').

If, say, `y1` and/or `x1` are ordered factors, the default scores, `1:nlevels(y1)` and `1:nlevels(x1)`, respectively, can be altered using the `scores` argument; this argument can also be used to coerce nominal factors to class `"ordered"`. For example, when `y1` is an ordered factor with four levels and `x1` is a nominal factor with three levels, `scores = list(y1 = c(1, 3:5), x1 = c(1:2, 4))` supplies the scores to be used. For ordered alternatives the scores must be monotonic, but non-monotonic scores are also allowed for testing against, e.g., umbrella alternatives. The length of the score vector must be equal to the number of factor levels.

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (`distribution = "asymptotic"`). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for univariate two-sample problems by setting `distribution` to `"approximate"` or `"exact"`, respectively. See `asymptotic()`, `approximate()` and `exact()` for details.

The example section uses data presented by Johnson, Mercante, and May (1993).

Value

An object inheriting from class `"IndependenceTest"`.

Note

Starting with version 1.1-0, maximum statistics and quadratic forms can no longer be specified using `teststat = "maxtype"` and `teststat = "quadtype"`, respectively (as was used in versions prior to 0.4-5).

References

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2006). "A Lego System for Conditional Inference." *The American Statistician*, **60**(3), 257–263. doi:10.1198/000313006X118430.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2008). "Implementing a Class of Permutation Tests: The **coin** Package." *Journal of Statistical Software*, **28**(8), 1–23. doi:10.18637/jss.v028.i08.

Johnson WD, Mercante DE, May WL (1993). "A Computer Package for the Multivariate Nonparametric Rank Test in Completely Randomized Experimental Designs." *Computer Methods and Programs in Biomedicine*, **40**(3), 217–225. ISSN 0169-2607. doi:10.1016/01692607(93)90059T.

Puri ML, Sen PK (1966). "On a Class of Multivariate Multisample Rank-Order Tests." *Sankhyā: The Indian Journal of Statistics, Series A*, **28**(4), 353–376.

Puri ML, Sen PK (1971). *Nonparametric Methods in Multivariate Analysis*. John Wiley & Sons, New York, U.S.A.

Strasser H, Weber C (1999). "On the Asymptotic Theory of Permutation Statistics." *Mathematical Methods of Statistics*, **8**(2), 220–250.

Examples

```
## One-sided exact van der Waerden (normal scores) test...
independence_test(asat ~ group, data = asat,
                  ## exact null distribution
                  distribution = "exact",
                  ## one-sided test
                  alternative = "greater",
                  ## apply normal scores to asat$asat
                  ytrafo = function(data)
                      trafo(data, numeric_trafo = normal_trafo),
                  ## indicator matrix of 1st level of asat$group
                  xtrafo = function(data)
                      trafo(data, factor_trafo = function(x)
                          matrix(x == levels(x)[1], ncol = 1)))

## ...or more conveniently
normal_test(asat ~ group, data = asat,
            ## exact null distribution
            distribution = "exact",
            ## one-sided test
            alternative = "greater")


## Receptor binding assay of benzodiazepines
## Johnson, Mercante and May (1993, Tab. 1)
benzos <- data.frame(
      cerebellum = c( 3.41,  3.50,  2.85,  4.43,
                      4.04,  7.40,  5.63, 12.86,
                      6.03,  6.08,  5.75,  8.09,  7.56),
       brainstem = c( 3.46,  2.73,  2.22,  3.16,
                      2.59,  4.18,  3.10,  4.49,
                      6.78,  7.54,  5.29,  4.57,  5.39),
          cortex = c(10.52,  7.52,  4.57,  5.48,
                      7.16, 12.00,  9.36,  9.35,
                     11.54, 11.05,  9.92, 13.59, 13.21),
    hypothalamus = c(19.51, 10.00,  8.27, 10.26,
                     11.43, 19.13, 14.03, 15.59,
                     24.87, 14.16, 22.68, 19.93, 29.32),
        striatum = c( 6.98,  5.07,  3.57,  5.34,
                      4.57,  8.82,  5.76, 11.72,
                      6.98,  7.54,  7.66,  9.69,  8.09),
     hippocampus = c(20.31, 13.20,  8.58, 11.42,
                     13.79, 23.71, 18.35, 38.52,
                     21.56, 18.66, 19.24, 27.39, 26.55),
       treatment = factor(rep(c("Lorazepam", "Alprazolam", "Saline"),
                           c(4, 4, 5)))
)

## Approximative (Monte Carlo) multivariate Kruskal-Wallis test
## Johnson, Mercante and May (1993, Tab. 2)
independence_test(cerebellum + brainstem + cortex +
                  hypothalamus + striatum + hippocampus ~ treatment,
```

```
    data = benzos,
    teststat = "quadratic",
    distribution = approximate(nresample = 10000),
    ytrafo = function(data)
        trafo(data, numeric_trafo = rank_trafo)) # Q = 16.129
```

IndependenceTest-class

Class "IndependenceTest" *and Its Subclasses*

Description

Objects of class "IndependenceTest" and its subclasses "MaxTypeIndependenceTest", "QuadTypeIndependenceTest", "ScalarIndependenceTest" and "ScalarIndependenceTestConfint" represent an independence test including its original and transformed data structure, linear statistic, test statistic and reference distribution.

Objects from the Class

Objects can be created by calls of the form

```
new("IndependenceTest", ...),

new("MaxTypeIndependenceTest", ...),

new("QuadTypeIndependenceTest", ...),

new("ScalarIndependenceTest", ...)
```

and

```
new("ScalarIndependenceTestConfint", ...).
```

Slots

For objects of classes "IndependenceTest", "MaxTypeIndependenceTest", "QuadTypeIndependenceTest", "ScalarIndependenceTest" or "ScalarIndependenceTestConfint":

`distribution`**:** Object of class "PValue". The reference distribution.

`statistic`**:** Object of class "IndependenceTestStatistic". The test statistic, the linear statistic, and the transformed and original data structures.

`estimates`**:** Object of class "list". The estimated parameters.

`method`**:** Object of class "character". The test method.

`call`**:** Object of class "call". The matched call.

Additionally, for objects of classes "ScalarIndependenceTest" or "ScalarIndependenceTestConfint":

`parameter`: Object of class `"character"`. The tested parameter.

`nullvalue`: Object of class `"numeric"`. The hypothesized value of the null hypothesis.

Additionally, for objects of class `"ScalarIndependenceTestConfint"`:

`confint`: Object of class `"function"`. The confidence interval function.

`conf.level`: Object of class `"numeric"`. The confidence level.

Extends

For objects of classes `"MaxTypeIndependenceTest"`, `"QuadTypeIndependenceTest"` or `"ScalarIndependenceTest"`:
Class `"IndependenceTest"`, directly.

For objects of class `"ScalarIndependenceTestConfint"`:
Class `"ScalarIndependenceTest"`, directly.
Class `"IndependenceTest"`, by class `"ScalarIndependenceTest"`, distance 2.

Known Subclasses

For objects of class `"IndependenceTest"`:
Class `"MaxTypeIndependenceTest"`, directly.
Class `"QuadTypeIndependenceTest"`, directly.
Class `"ScalarIndependenceTest"`, directly.
Class `"ScalarIndependenceTestConfint"`, by class `"ScalarIndependenceTest"`, distance 2.

For objects of class `"ScalarIndependenceTest"`:
Class `"ScalarIndependenceTestConfint"`, directly.

Methods

confint `signature(object = "IndependenceTest")`: See the documentation for `confint-methods` (in package **stats4**) for details.

confint `signature(object = "ScalarIndependenceTestConfint")`: See the documentation for `confint-methods` (in package **stats4**) for details.

covariance `signature(object = "IndependenceTest")`: See the documentation for `covariance()` for details.

dperm `signature(object = "IndependenceTest")`: See the documentation for `dperm()` for details.

expectation `signature(object = "IndependenceTest")`: See the documentation for `expectation()` for details.

midpvalue `signature(object = "IndependenceTest")`: See the documentation for `midpvalue()` for details.

pperm `signature(object = "IndependenceTest")`: See the documentation for `pperm()` for details.

pvalue `signature(object = "IndependenceTest")`: See the documentation for `pvalue()` for details.

pvalue `signature(object = "MaxTypeIndependenceTest")`: See the documentation for `pvalue()` for details.

pvalue_interval `signature(object = "IndependenceTest")`: See the documentation for `pvalue_interval()` for details.

qperm `signature(object = "IndependenceTest")`: See the documentation for `qperm()` for details.

rperm `signature(object = "IndependenceTest")`: See the documentation for `rperm()` for details.

show `signature(object = "IndependenceTest")`: See the documentation for `show()` (in package **methods**) for details.

show `signature(object = "MaxTypeIndependenceTest")`: See the documentation for `show()` (in package **methods**) for details.

show `signature(object = "QuadTypeIndependenceTest")`: See the documentation for `show()` (in package **methods**) for details.

show `signature(object = "ScalarIndependenceTest")`: See the documentation for `show()` (in package **methods**) for details.

show `signature(object = "ScalarIndependenceTestConfint")`: See the documentation for `show()` (in package **methods**) for details.

size `signature(object = "IndependenceTest")`: See the documentation for `size()` for details.

statistic `signature(object = "IndependenceTest")`: See the documentation for `statistic()` for details.

support `signature(object = "IndependenceTest")`: See the documentation for `support()` for details.

variance `signature(object = "IndependenceTest")`: See the documentation for `variance()` for details.

`IndependenceTestProblem-class`

Class `"IndependenceTestProblem"`

Description

Objects of class `"IndependenceTestProblem"` represent the transformed and original data structures corresponding to an independence problem.

Objects from the Class

Objects can be created by calls of the form

```
new("IndependenceTestProblem", object, xtrafo = trafo, ytrafo = trafo, ...)
```

where `object` is an object of class `"IndependenceProblem"`, `xtrafo` is the regression function $g(\mathbf{X})$ and `ytrafo` is the influence function $h(\mathbf{Y})$.

Slots

`xtrans`: Object of class `"matrix"`. The transformed `x`.

`ytrans`: Object of class `"matrix"`. The transformed `y`.

`xtrafo`: Object of class `"function"`. The regression function for `x`.

`ytrafo`: Object of class `"function"`. The influence function for `y`.

`x`: Object of class `"data.frame"`. The variables `x`.

`y`: Object of class `"data.frame"`. The variables `y`.

`block`: Object of class `"factor"`. The block structure.

`weights`: Object of class `"numeric"`. The case weights.

Extends

Class `"IndependenceProblem"`, directly.

Known Subclasses

Class `"IndependenceLinearStatistic"`, directly.
Class `"IndependenceTestStatistic"`, by class `"IndependenceLinearStatistic"`, distance 2.
Class `"MaxTypeIndependenceTestStatistic"`, by class `"IndependenceTestStatistic"`, distance 3.
Class `"QuadTypeIndependenceTestStatistic"`, by class `"IndependenceTestStatistic"`, distance 3.
Class `"ScalarIndependenceTestStatistic"`, by class `"IndependenceTestStatistic"`, distance 3.

Methods

initialize `signature(.Object = "IndependenceTestProblem")`: See the documentation for `initialize()` (in package **methods**) for details.

IndependenceTestStatistic-class

Class `"IndependenceTestStatistic"` *and Its Subclasses*

Description

Objects of class `"IndependenceTestStatistic"` and its subclasses `"MaxTypeIndependenceTestStatistic"`, `"QuadTypeIndependenceTestStatistic"` and `"ScalarIndependenceTestStatistic"` represent the test statistic, the linear statistic, and the transformed and original data structures corresponding to an independence problem.

Objects from the Class

Class "IndependenceTestStatistic" is a *virtual* class, so objects cannot be created from it directly.

Objects can be created by calls of the form

```
new("MaxTypeIndependenceTestStatistic", object,
    alternative = c("two.sided", "less", "greater"), ...),

new("QuadTypeIndependenceTestStatistic", object, paired = FALSE, ...)
```

and

```
new("ScalarIndependenceTestStatistic", object,
    alternative = c("two.sided", "less", "greater"), paired = FALSE, ...)
```

where `object` is an object of class "IndependenceLinearStatistic", `alternative` is a character specifying the direction of the alternative hypothesis and `paired` is a logical indicating that paired data have been transformed in such a way that the (unstandardized) linear statistic is the sum of the absolute values of the positive differences between the paired observations.

Slots

For objects of classes "IndependenceTestStatistic", "MaxTypeIndependenceTestStatistic", "QuadTypeIndependenceTestStatistic" or "ScalarIndependenceTestStatistic":

`teststatistic`: Object of class "numeric". The test statistic.

`standardizedlinearstatistic`: Object of class "numeric". The standardized linear statistic.

`linearstatistic`: Object of class "matrix". The linear statistic for each block.

`expectation`: Object of class "matrix". The expectation of the linear statistic for each block.

`covariance`: Object of class "matrix". The lower triangular elements of the covariance of the linear statistic for each block.

`xtrans`: Object of class "matrix". The transformed `x`.

`ytrans`: Object of class "matrix". The transformed `y`.

`xtrafo`: Object of class "function". The regression function for `x`.

`ytrafo`: Object of class "function". The influence function for `y`.

`x`: Object of class "data.frame". The variables `x`.

`y`: Object of class "data.frame". The variables `y`.

`block`: Object of class "factor". The block structure.

`weights`: Object of class "numeric". The case weights.

Additionally, for objects of classes "MaxTypeIndependenceTest" or "ScalarIndependenceTest":

`alternative`: Object of class "character". The direction of the alternative hypothesis.

Additionally, for objects of class "QuadTypeIndependenceTest":

`covarianceplus`: Object of class `"numeric"`. The lower triangular elements of the Moore-Penrose inverse of the covariance of the linear statistic.

`df`: Object of class `"numeric"`. The rank of the covariance matrix.

Additionally, for objects of classes `"QuadTypeIndependenceTest"` or `"ScalarIndependenceTest"`:

`paired`: Object of class `"logical"`. The indicator for paired test statistics.

Extends

For objects of class `"IndependenceTestStatistic"`:
Class `"IndependenceLinearStatistic"`, directly.
Class `"IndependenceTestProblem"`, by class `"IndependenceLinearStatistic"`, distance 2.
Class `"IndependenceProblem"`, by class `"IndependenceLinearStatistic"`, distance 3.

For objects of classes `"MaxTypeIndependenceTestStatistic"`, `"QuadTypeIndependenceTestStatistic"` or `"ScalarIndependenceTestStatistic"`:
Class `"IndependenceTestStatistic"`, directly.
Class `"IndependenceLinearStatistic"`, by class `"IndependenceTestStatistic"`, distance 2.
Class `"IndependenceTestProblem"`, by class `"IndependenceTestStatistic"`, distance 3.
Class `"IndependenceProblem"`, by class `"IndependenceTestStatistic"`, distance 4.

Known Subclasses

For objects of class `"IndependenceTestStatistic"`:
Class `"MaxTypeIndependenceTestStatistic"`, directly.
Class `"QuadTypeIndependenceTestStatistic"`, directly.
Class `"ScalarIndependenceTestStatistic"`, directly.

Methods

ApproxNullDistribution `signature(object = "MaxTypeIndependenceTestStatistic")`: See the documentation for `ApproxNullDistribution()` for details.

ApproxNullDistribution `signature(object = "QuadTypeIndependenceTestStatistic")`: See the documentation for `ApproxNullDistribution()` for details.

ApproxNullDistribution `signature(object = "ScalarIndependenceTestStatistic")`: See the documentation for `ApproxNullDistribution()` for details.

AsymptNullDistribution `signature(object = "MaxTypeIndependenceTestStatistic")`: See the documentation for `AsymptNullDistribution()` for details.

AsymptNullDistribution `signature(object = "QuadTypeIndependenceTestStatistic")`: See the documentation for `AsymptNullDistribution()` for details.

AsymptNullDistribution `signature(object = "ScalarIndependenceTestStatistic")`: See the documentation for `AsymptNullDistribution()` for details.

ExactNullDistribution `signature(object = "QuadTypeIndependenceTestStatistic")`: See the documentation for `ExactNullDistribution()` for details.

ExactNullDistribution `signature(object = "ScalarIndependenceTestStatistic")`: See the documentation for `ExactNullDistribution()` for details.

covariance `signature(object = "QuadTypeIndependenceTestStatistic")`: See the documentation for `covariance()` for details.

initialize `signature(.Object = "IndependenceTestStatistic")`: See the documentation for `initialize()` (in package **methods**) for details.

initialize `signature(.Object = "MaxTypeIndependenceTestStatistic")`: See the documentation for `initialize()` (in package **methods**) for details.

initialize `signature(.Object = "QuadTypeIndependenceTestStatistic")`: See the documentation for `initialize()` (in package **methods**) for details.

initialize `signature(.Object = "ScalarIndependenceTestStatistic")`: See the documentation for `initialize()` (in package **methods**) for details.

statistic `signature(object = "IndependenceTestStatistic")`: See the documentation for `statistic()` for details.

`jobsatisfaction` *Income and Job Satisfaction*

Description

Income and job satisfaction by gender.

Usage

```
jobsatisfaction
```

Format

A contingency table with 104 observations on 3 variables.

`Income` a factor with levels `"<5000"`, `"5000-15000"`, `"15000-25000"` and `">25000"`.

`Job.Satisfaction` a factor with levels `"Very Dissatisfied"`, `"A Little Satisfied"`, `"Moderately Satisfied"` and `"Very Satisfied"`.

`Gender` a factor with levels `"Female"` and `"Male"`.

Details

This data set was given in Agresti (2002, Tab. 7.8 on p. 288). Winell and Lindbäck (2018) used the data to demonstrate a score-independent test for ordered categorical data.

References

Agresti A (2002). *Categorical Data Analysis*, 2nd edition. John Wiley & Sons, Hoboken, New Jersey.

Winell H, Lindbäck J (2018). "A General Score-independent Test For Order-restricted Inference." *Statistics in Medicine*, **37**(21), 3078–3090. doi:10.1002/sim.7690.

Examples

```
## Approximative (Monte Carlo) linear-by-linear association test
lbl_test(jobsatisfaction, distribution = approximate(nresample = 10000))

## Not run:
## Approximative (Monte Carlo) score-independent test
## Winell and Lindbaeck (2018)
(it <- independence_test(jobsatisfaction,
                         distribution = approximate(nresample = 10000),
                         xtrafo = function(data)
                             trafo(data, factor_trafo = function(x)
                                 zheng_trafo(as.ordered(x))),
                         ytrafo = function(data)
                             trafo(data, factor_trafo = function(y)
                                 zheng_trafo(as.ordered(y)))))

## Extract the "best" set of scores
ss <- statistic(it, type = "standardized")
idx <- which(abs(ss) == max(abs(ss)), arr.ind = TRUE)
ss[idx[1], idx[2], drop = FALSE]
## End(Not run)
```

LocationTests *Two- and K-Sample Location Tests*

Description

Testing the equality of the distributions of a numeric response variable in two or more independent groups against shift alternatives.

Usage

```
## S3 method for class 'formula'
oneway_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
oneway_test(object, ...)

## S3 method for class 'formula'
wilcox_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
wilcox_test(object, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
kruskal_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
kruskal_test(object, ...)

## S3 method for class 'formula'
```

```
normal_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
normal_test(object, ties.method = c("mid-ranks", "average-scores"),
            conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
median_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
median_test(object, mid.score = c("0", "0.5", "1"),
            conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
savage_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
savage_test(object, ties.method = c("mid-ranks", "average-scores"),
            conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

formula	a formula of the form y ~ x | block where y is a numeric variable, x is a factor and block is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations.
object	an object inheriting from class "IndependenceProblem".
conf.int	a logical indicating whether a confidence interval for the difference in location should be computed. Defaults to FALSE.
conf.level	a numeric, confidence level of the interval. Defaults to 0.95.
ties.method	a character, the method used to handle ties: the score generating function either uses mid-ranks ("mid-ranks", default) or averages the scores of randomly broken ties ("average-scores").
mid.score	a character, the score assigned to observations exactly equal to the median: either 0 ("0", default), 0.5 ("0.5") or 1 ("1"); see 'Details'.
...	further arguments to be passed to independence_test().

Details

oneway_test(), wilcox_test(), kruskal_test(), normal_test(), median_test() and savage_test() provide the Fisher-Pitman permutation test, the Wilcoxon-Mann-Whitney test, the Kruskal-Wallis test, the van der Waerden test, the Brown-Mood median test and the Savage test. A general description of these methods is given by Hollander and Wolfe (1999).For the adjustment of scores for tied values see Hájek, Šidák, and Sen (1999, pp. 133–135).

The null hypothesis of equality, or conditional equality given block, of the distribution of y in the groups defined by x is tested against shift alternatives. In the two-sample case, the two-sided null

hypothesis is $H_0 : \mu = 0$, where $\mu = Y_1 - Y_2$ and Y_s is the median of the responses in the sth sample. In case `alternative = "less"`, the null hypothesis is $H_0 : \mu \geq 0$. When `alternative = "greater"`, the null hypothesis is $H_0 : \mu \leq 0$. Confidence intervals for the difference in location are available (except for `oneway_test()`) and computed according to Bauer (1972).

If `x` is an ordered factor, the default scores, `1:nlevels(x)`, can be altered using the `scores` argument (see `independence_test()`); this argument can also be used to coerce nominal factors to class `"ordered"`. In this case, a linear-by-linear association test is computed and the direction of the alternative hypothesis can be specified using the `alternative` argument.

The Brown-Mood median test offers a choice of mid-score, i.e., the score assigned to observations exactly equal to the median. In the two-sample case, `mid-score = "0"` implies that the linear test statistic is simply the number of subjects in the second sample with observations greater than the median of the pooled sample. Similarly, the linear test statistic for the last alternative, `mid-score = "1"`, is the number of subjects in the second sample with observations greater than or equal to the median of the pooled sample. If `mid-score = "0.5"` is selected, the linear test statistic is the mean of the test statistics corresponding to the first and last alternatives and has a symmetric distribution, or at least approximately so, under the null hypothesis (see Hájek et al. 1999, pp. 97–98).

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (`distribution = "asymptotic"`). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for univariate two-sample problems by setting `distribution` to `"approximate"` or `"exact"`, respectively. See `asymptotic()`, `approximate()` and `exact()` for details.

Value

An object inheriting from class `"IndependenceTest"`. Confidence intervals can be extracted by `confint()`.

Note

Starting with version 1.1-0, `oneway_test()` no longer allows the test statistic to be specified; a quadratic form is now used in the K-sample case. Please use `independence_test()` if more control is desired.

References

Bauer DF (1972). "Constructing Confidence Sets Using Rank Statistics." *Journal of the American Statistical Association*, 687–690. doi:10.1080/01621459.1972.10481279.

Hájek J, Šidák Z, Sen PK (1999). *Theory of Rank Tests*, 2nd edition. Academic Press, London, UK.

Hollander M, Wolfe DA (1999). *Nonparametric Statistical Inference*, 2nd edition. John Wiley & Sons, New York, U.S.A.

Examples

```
## Tritiated Water Diffusion Across Human Chorioamnion
## Hollander and Wolfe (1999, p. 110, Tab. 4.1)
diffusion <- data.frame(
    pd = c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46,
           1.15, 0.88, 0.90, 0.74, 1.21),
```

```
    age = factor(rep(c("At term", "12-26 Weeks"), c(10, 5)))
)

## Exact Wilcoxon-Mann-Whitney test
## Hollander and Wolfe (1999, p. 111)
## (At term - 12-26 Weeks)
(wt <- wilcox_test(pd ~ age, data = diffusion,
                   distribution = "exact", conf.int = TRUE))

## Extract observed Wilcoxon statistic
## Note: this is the sum of the ranks for age = "12-26 Weeks"
statistic(wt, type = "linear")

## Expectation, variance, two-sided pvalue and confidence interval
expectation(wt)
covariance(wt)
pvalue(wt)
confint(wt)

## For two samples, the Kruskal-Wallis test is equivalent to the W-M-W test
kruskal_test(pd ~ age, data = diffusion,
             distribution = "exact")

## Asymptotic Fisher-Pitman test
oneway_test(pd ~ age, data = diffusion)

## Approximative (Monte Carlo) Fisher-Pitman test
pvalue(oneway_test(pd ~ age, data = diffusion,
                   distribution = approximate(nresample = 10000)))

## Exact Fisher-Pitman test
pvalue(ot <- oneway_test(pd ~ age, data = diffusion,
                         distribution = "exact"))

## Plot density and distribution of the standardized test statistic
op <- par(no.readonly = TRUE) # save current settings
layout(matrix(1:2, nrow = 2))
s <- support(ot)
d <- dperm(ot, s)
p <- pperm(ot, s)
plot(s, d, type = "S", xlab = "Test Statistic", ylab = "Density")
plot(s, p, type = "S", xlab = "Test Statistic", ylab = "Cum. Probability")
par(op) # reset


## Example data
ex <- data.frame(
    y = c(3, 4, 8, 9, 1, 2, 5, 6, 7),
    x = factor(rep(c("no", "yes"), c(4, 5)))
)

## Boxplots
boxplot(y ~ x, data = ex)
```

```
## Exact Brown-Mood median test with different mid-scores
(mt1 <- median_test(y ~ x, data = ex, distribution = "exact"))
(mt2 <- median_test(y ~ x, data = ex, distribution = "exact",
                    mid.score = "0.5"))
(mt3 <- median_test(y ~ x, data = ex, distribution = "exact",
                    mid.score = "1")) # sign change!

## Plot density and distribution of the standardized test statistics
op <- par(no.readonly = TRUE) # save current settings
layout(matrix(1:3, nrow = 3))
s1 <- support(mt1); d1 <- dperm(mt1, s1)
plot(s1, d1, type = "h", main = "Mid-score: 0",
     xlab = "Test Statistic", ylab = "Density")
s2 <- support(mt2); d2 <- dperm(mt2, s2)
plot(s2, d2, type = "h", main = "Mid-score: 0.5",
     xlab = "Test Statistic", ylab = "Density")
s3 <- support(mt3); d3 <- dperm(mt3, s3)
plot(s3, d3, type = "h", main = "Mid-score: 1",
     xlab = "Test Statistic", ylab = "Density")
par(op) # reset


## Length of YOY Gizzard Shad
## Hollander and Wolfe (1999, p. 200, Tab. 6.3)
yoy <- data.frame(
    length = c(46, 28, 46, 37, 32, 41, 42, 45, 38, 44,
               42, 60, 32, 42, 45, 58, 27, 51, 42, 52,
               38, 33, 26, 25, 28, 28, 26, 27, 27, 27,
               31, 30, 27, 29, 30, 25, 25, 24, 27, 30),
    site = gl(4, 10, labels = as.roman(1:4))
)

## Approximative (Monte Carlo) Kruskal-Wallis test
kruskal_test(length ~ site, data = yoy,
             distribution = approximate(nresample = 10000))

## Approximative (Monte Carlo) Nemenyi-Damico-Wolfe-Dunn test (joint ranking)
## Hollander and Wolfe (1999, p. 244)
## (where Steel-Dwass results are given)
it <- independence_test(length ~ site, data = yoy,
                        distribution = approximate(nresample = 50000),
                        ytrafo = function(data)
                            trafo(data, numeric_trafo = rank_trafo),
                        xtrafo = mcp_trafo(site = "Tukey"))

## Global p-value
pvalue(it)

## Sites (I = II) != (III = IV) at alpha = 0.01 (p. 244)
pvalue(it, method = "single-step") # subset pivotality is violated
```

`malformations` *Maternal Drinking and Congenital Sex Organ Malformation*

Description

A subset of data from a study on the relationship between maternal alcohol consumption and congenital malformations.

Usage

```
malformations
```

Format

A data frame with 32574 observations on 2 variables.

`consumption` alcohol consumption, an ordered factor with levels `"0"`, `"<1"`, `"1-2"`, `"3-5"` and `">=6"`.

`malformation` congenital sex organ malformation, a factor with levels `"Present"` and `"Absent"`.

Details

Data from a prospective study undertaken to determine whether moderate or light drinking during the first trimester of pregnancy increases the risk for congenital malformations (Mills and Graubard 1987). The subset given here concerns only sex organ malformation (Mills and Graubard 1987, Tab. 4).

This data set was used by Graubard and Korn (1987) to illustrate that different choices of scores for ordinal variables can lead to conflicting conclusions. Zheng (2008) also used the data, demonstrating two different score-independent tests for ordered categorical data; see also Winell and Lindbäck (2018).

References

Graubard BI, Korn EL (1987). "Choice of Column Scores for Testing Independence in Ordered $2 \times K$ Contingency Tables." *Biometrics*, **43**(2), 471–476. doi:10.2307/2531828.

Mills JL, Graubard BI (1987). "Is Moderate Drinking During Pregnancy Associated With an Increased Risk for Malformations?" *Pediatrics*, **80**(3), 309–314. doi:10.1542/peds.80.3.309.

Winell H, Lindbäck J (2018). "A General Score-independent Test For Order-restricted Inference." *Statistics in Medicine*, **37**(21), 3078–3090. doi:10.1002/sim.7690.

Zheng G (2008). "Analysis of Ordered Categorical Data: Two Score-Independent Approaches." *Biometrics*, **64**(4), 1276–1279. doi:10.1111/j.15410420.2008.00992.x.

Examples

```
## Graubard and Korn (1987, Tab. 3)

## One-sided approximative (Monte Carlo) Cochran-Armitage test
## Note: midpoint scores (p < 0.05)
midpoints <- c(0, 0.5, 1.5, 4.0, 7.0)
chisq_test(malformation ~ consumption, data = malformations,
           distribution = approximate(nresample = 1000),
           alternative = "greater",
           scores = list(consumption = midpoints))

## One-sided approximative (Monte Carlo) Cochran-Armitage test
## Note: midrank scores (p > 0.05)
midranks <- c(8557.5, 24375.5, 32013.0, 32473.0, 32555.5)
chisq_test(malformation ~ consumption, data = malformations,
           distribution = approximate(nresample = 1000),
           alternative = "greater",
           scores = list(consumption = midranks))

## One-sided approximative (Monte Carlo) Cochran-Armitage test
## Note: equally spaced scores (p > 0.05)
chisq_test(malformation ~ consumption, data = malformations,
           distribution = approximate(nresample = 1000),
           alternative = "greater")

## Not run:
## One-sided approximative (Monte Carlo) score-independent test
## Winell and Lindbaeck (2018)
(it <- independence_test(malformation ~ consumption, data = malformations,
                         distribution = approximate(nresample = 1000,
                                                    parallel = "snow",
                                                    ncpus = 8),
                         alternative = "greater",
                         xtrafo = function(data)
                             trafo(data, ordered_trafo = zheng_trafo)))

## Extract the "best" set of scores
ss <- statistic(it, type = "standardized")
idx <- which(ss == max(ss), arr.ind = TRUE)
ss[idx[1], idx[2], drop = FALSE]
## End(Not run)
```

MarginalHomogeneityTests

Marginal Homogeneity Tests

Description

Testing the marginal homogeneity of a repeated measurements factor in a complete block design.

Usage

```
## S3 method for class 'formula'
mh_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
mh_test(object, ...)
## S3 method for class 'SymmetryProblem'
mh_test(object, ...)
```

Arguments

formula	a formula of the form y ~ x | block where y and x are factors and block is an optional factor (which is generated automatically if omitted).
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations. (Not yet implemented!)
object	an object inheriting from classes "table" (with identical dimnames components) or "SymmetryProblem".
...	further arguments to be passed to symmetry_test().

Details

mh_test() provides the McNemar test, the Cochran Q test, the Stuart(-Maxwell) test and the Madansky test of interchangeability. A general description of these methods is given by Agresti (2002).

The null hypothesis of marginal homogeneity is tested. The response variable and the measurement conditions are given by y and x, respectively, and block is a factor where each level corresponds to exactly one subject with repeated measurements.

This procedure is known as the McNemar test (McNemar 1947) when both y and x are binary factors, as the Cochran Q test (Cochran 1954) when y is a binary factor and x is a factor with an arbitrary number of levels, as the Stuart(-Maxwell) test (Stuart 1955; Maxwell 1970) when y is a factor with an arbitrary number of levels and x is a binary factor, and as the Madansky test of interchangeability (Madansky 1963), which implies marginal homogeneity, when both y and x are factors with an arbitrary number of levels.

If y and/or x are ordered factors, the default scores, 1:nlevels(y) and 1:nlevels(x), respectively, can be altered using the scores argument (see symmetry_test()); this argument can also be used to coerce nominal factors to class "ordered". If both y and x are ordered factors, a linear-by-linear association test is computed and the direction of the alternative hypothesis can be specified using the alternative argument. This extension was given by Birch (1965) who also discussed the situation when either the response or the measurement condition is an ordered factor; see also White, Landis, and Cooper (1982).

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (distribution = "asymptotic"). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for

univariate two-sample problems by setting `distribution` to `"approximate"` or `"exact"`, respectively. See `asymptotic()`, `approximate()` and `exact()` for details.

Value

An object inheriting from class `"IndependenceTest"`.

Note

This function is currently computationally inefficient for data with a large number of pairs or sets.

References

Agresti A (2002). *Categorical Data Analysis*, 2nd edition. John Wiley & Sons, Hoboken, New Jersey.

Birch MW (1965). "The Detection of Partial Association, II: The General Case." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **27**(1), 111–124. doi:10.1111/j.2517-6161.1965.tb00593.x.

Cochran WG (1954). "Some Methods for Strengthening the Common χ^2 Tests." *Biometrics*, **10**(4), 417–451. doi:10.2307/3001616.

Madansky A (1963). "Tests of Homogeneity for Correlated Samples." *Journal of the American Statistical Association*, **58**(301), 97–119. doi:10.1080/01621459.1963.10500835.

Maxwell AE (1970). "Comparing the Classification of Subjects by Two Independent Judges." *British Journal of Psychiatry*, **116**(535), 651–655. doi:10.1192/bjp.116.535.651.

McNemar Q (1947). "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages." *Psychometrika*, **12**(2), 153–157. doi:10.1007/bf02295996.

Stuart A (1955). "A Test for Homogeneity of the Marginal Distributions in a Two-way Classification." *Biometrika*, **42**(3–4), 412–416. doi:10.1093/biomet/42.34.412.

White AA, Landis JR, Cooper MM (1982). "A Note on the Equivalence of Several Marginal Homogeneity Test Criteria for Categorical Data." *International Statistical Review / Revue Internationale de Statistique*, **50**(1), 27–34. doi:10.2307/1402457.

Examples

```
## Performance of prime minister
## Agresti (2002, p. 409)
performance <- matrix(
    c(794, 150,
       86, 570),
    nrow = 2, byrow = TRUE,
    dimnames = list(
         "First" = c("Approve", "Disprove"),
        "Second" = c("Approve", "Disprove")
    )
)
performance <- as.table(performance)
diag(performance) <- 0 # speed-up: only off-diagonal elements contribute
```

```
## Asymptotic McNemar Test
mh_test(performance)

## Exact McNemar Test
mh_test(performance, distribution = "exact")


## Effectiveness of different media for the growth of diphtheria
## Cochran (1950, Tab. 2)
cases <- c(4, 2, 3, 1, 59)
n <- sum(cases)
cochran <- data.frame(
    diphtheria = factor(
        unlist(rep(list(c(1, 1, 1, 1),
                        c(1, 1, 0, 1),
                        c(0, 1, 1, 1),
                        c(0, 1, 0, 1),
                        c(0, 0, 0, 0)),
                   cases))
    ),
    media = factor(rep(LETTERS[1:4], n)),
    case =  factor(rep(seq_len(n), each = 4))
)

## Asymptotic Cochran Q test (Cochran, 1950, p. 260)
mh_test(diphtheria ~ media | case, data = cochran) # Q = 8.05

## Approximative Cochran Q test
mt <- mh_test(diphtheria ~ media | case, data = cochran,
              distribution = approximate(nresample = 10000))
pvalue(mt)             # standard p-value
midpvalue(mt)          # mid-p-value
pvalue_interval(mt)    # p-value interval
size(mt, alpha = 0.05) # test size at alpha = 0.05 using the p-value


## Opinions on Pre- and Extramarital Sex
## Agresti (2002, p. 421)
opinions <- c("Always wrong", "Almost always wrong",
              "Wrong only sometimes", "Not wrong at all")
PreExSex <- matrix(
    c(144, 33, 84, 126,
        2,  4, 14,  29,
        0,  2,  6,  25,
        0,  0,  1,   5),
    nrow = 4,
    dimnames = list(
          "Premarital Sex" = opinions,
        "Extramarital Sex" = opinions
    )
)
PreExSex <- as.table(PreExSex)
```

```
## Asymptotic Stuart test
mh_test(PreExSex)

## Asymptotic Stuart-Birch test
## Note: response as ordinal
mh_test(PreExSex, scores = list(response = 1:length(opinions)))


## Vote intention
## Madansky (1963, pp. 107-108)
vote <- array(
    c(120, 1,  8, 2,   2,  1, 2, 1,  7,
        6, 2,  1, 1, 103,  5, 1, 4,  8,
       20, 3, 31, 1,   6, 30, 2, 1, 81),
    dim = c(3, 3, 3),
    dimnames = list(
          "July" = c("Republican", "Democratic", "Uncertain"),
        "August" = c("Republican", "Democratic", "Uncertain"),
          "June" = c("Republican", "Democratic", "Uncertain")
    )
)
vote <- as.table(vote)

## Asymptotic Madansky test (Q = 70.77)
mh_test(vote)


## Cross-over study
## http://www.nesug.org/proceedings/nesug00/st/st9005.pdf
dysmenorrhea <- array(
    c(6, 2, 1,  3, 1, 0,  1, 2, 1,
      4, 3, 0, 13, 3, 0,  8, 1, 1,
      5, 2, 2, 10, 1, 0, 14, 2, 0),
    dim = c(3, 3, 3),
    dimnames =  list(
          "Placebo" = c("None", "Moderate", "Complete"),
         "Low dose" = c("None", "Moderate", "Complete"),
        "High dose" = c("None", "Moderate", "Complete")
    )
)
dysmenorrhea <- as.table(dysmenorrhea)

## Asymptotic Madansky-Birch test (Q = 53.76)
## Note: response as ordinal
mh_test(dysmenorrhea, scores = list(response = 1:3))

## Asymptotic Madansky-Birch test (Q = 47.29)
## Note: response and measurement conditions as ordinal
mh_test(dysmenorrhea, scores = list(response = 1:3,
                                    conditions = 1:3))
```

MaximallySelectedStatisticsTests
Generalized Maximally Selected Statistics

Description

Testing the independence of two sets of variables measured on arbitrary scales against cutpoint alternatives.

Usage

```
## S3 method for class 'formula'
maxstat_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
maxstat_test(object, ...)
## S3 method for class 'IndependenceProblem'
maxstat_test(object, teststat = c("maximum", "quadratic"),
             distribution = c("asymptotic", "approximate", "none"),
             minprob = 0.1, maxprob = 1 - minprob, ...)
```

Arguments

formula	a formula of the form y1 + ... + yq ~ x1 + ... + xp | block where y1, ..., yq and x1, ..., xp are measured on arbitrary scales (nominal, ordinal or continuous with or without censoring) and block is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations.
object	an object inheriting from classes "table" or "IndependenceProblem".
teststat	a character, the type of test statistic to be applied: either a maximum statistic ("maximum", default) or a quadratic form ("quadratic").
distribution	a character, the conditional null distribution of the test statistic can be approximated by its asymptotic distribution ("asymptotic", default) or via Monte Carlo resampling ("approximate"). Alternatively, the functions asymptotic or approximate can be used. Computation of the null distribution can be suppressed by specifying "none".
minprob	a numeric, a fraction between 0 and 0.5 specifying that cutpoints only greater than the minprob · 100% quantile of x1, ..., xp are considered. Defaults to 0.1.
maxprob	a numeric, a fraction between 0.5 and 1 specifying that cutpoints only smaller than the maxprob · 100% quantile of x1, ..., xp are considered. Defaults to 1 - minprob.
...	further arguments to be passed to independence_test().

Details

`maxstat_test()` provides generalized maximally selected statistics. The family of maximally selected statistics encompasses a large collection of procedures used for the estimation of simple cutpoint models including, but not limited to, maximally selected χ^2 statistics, maximally selected Cochran-Armitage statistics, maximally selected rank statistics and maximally selected statistics for multiple covariates. A general description of these methods is given by Hothorn and Zeileis (2008).

The null hypothesis of independence, or conditional independence given `block`, between `y1`, ..., `yq` and `x1`, ..., `xp` is tested against cutpoint alternatives. All possible partitions into two groups are evaluated for each unordered covariate `x1`, ..., `xp`, whereas only order-preserving binary partitions are evaluated for ordered or numeric covariates. The cutpoint is then a set of levels defining one of the two groups.

If both response and covariate is univariable, say `y1` and `x1`, this procedure is known as maximally selected χ^2 statistics (Miller and Siegmund 1982) when `y1` is a binary factor and `x1` is a numeric variable, and as maximally selected rank statistics when `y1` is a rank transformed numeric variable and `x1` is a numeric variable (Lausen and Schumacher 1992). Lausen, Hothorn, Bretz, and Schumacher (2004) introduced maximally selected statistics for a univariable numeric response and multiple numeric covariates `x1`, ..., `xp`.

If, say, `y1` and/or `x1` are ordered factors, the default scores, `1:nlevels(y1)` and `1:nlevels(x1)`, respectively, can be altered using the `scores` argument (see `independence_test()`); this argument can also be used to coerce nominal factors to class `"ordered"`. If both, say, `y1` and `x1` are ordered factors, a linear-by-linear association test is computed and the direction of the alternative hypothesis can be specified using the `alternative` argument. The particular extension to the case of a univariable ordered response and a univariable numeric covariate was given by Betensky and Rabinowitz (1999) and is known as maximally selected Cochran-Armitage statistics.

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (`distribution = "asymptotic"`). Alternatively, the distribution can be approximated via Monte Carlo resampling by setting `distribution` to `"approximate"`. See `asymptotic()` and `approximate()` for details.

The tree pipit data used in the example section was described by Müller and Hothorn (2004).

Value

An object inheriting from class `"IndependenceTest"`.

Note

Starting with version 1.1-0, maximum statistics and quadratic forms can no longer be specified using `teststat = "maxtype"` and `teststat = "quadtype"`, respectively (as was used in versions prior to 0.4-5).

References

Betensky RA, Rabinowitz D (1999). "Maximally Selected χ^2 Statistics for $k \times 2$ Tables." *Biometrics*, **55**(1), 317–320. doi:10.1111/j.0006341x.1999.00317.x.

Hothorn T, Zeileis A (2008). "Generalized Maximally Selected Statistics." *Biometrics*, **64**(4), 1263–1269. doi:10.1111/j.15410420.2008.00995.x.

Lausen B, Hothorn T, Bretz F, Schumacher M (2004). "Assessment of Optimal Selected Prognostic Factors." *Biometrical Journal*, **46**(3), 364–374.

Lausen B, Schumacher M (1992). "Maximally Selected Rank Statistics." *Biometrics*, **48**(1), 73–85. doi:10.2307/2532740.

Miller R, Siegmund D (1982). "Maximally Selected Chi Square Statistics." *Biometrics*, **38**(4), 1011–1016. doi:10.2307/2529881.

Müller J, Hothorn T (2004). "Maximally Selected Two-Sample Statistics as a New Tool for the Identification and Assessment of Habitat Factors with an Application to Breeding Bird Communities in Oak Forests." *European Journal of Forest Research*, **123**, 218–228. doi:10.1007/s10342004-00355.

Examples

```
## Tree pipit data (Mueller and Hothorn, 2004)
## Asymptotic maximally selected statistics
maxstat_test(counts ~ coverstorey, data = treepipit)

## Asymptotic maximally selected statistics
## Note: all covariates simultaneously
mt <- maxstat_test(counts ~ ., data = treepipit)
mt@estimates$estimate


## Malignant arrythmias data (Hothorn and Lausen, 2003, Sec. 7.2)
## Asymptotic maximally selected statistics
maxstat_test(Surv(time, event) ~  EF, data = hohnloser,
             ytrafo = function(data)
                 trafo(data, surv_trafo = function(y)
                     logrank_trafo(y, ties.method = "Hothorn-Lausen")))


## Breast cancer data (Hothorn and Lausen, 2003, Sec. 7.3)
## Asymptotic maximally selected statistics
if (requireNamespace("TH.data")) {
  data("sphase", package = "TH.data")
  maxstat_test(Surv(RFS, event) ~  SPF, data = sphase,
               ytrafo = function(data)
                   trafo(data, surv_trafo = function(y)
                       logrank_trafo(y, ties.method = "Hothorn-Lausen")))
}


## Job satisfaction data (Agresti, 2002, p. 288, Tab. 7.8)
## Asymptotic maximally selected statistics
maxstat_test(jobsatisfaction)

## Asymptotic maximally selected statistics
## Note: 'Job.Satisfaction' and 'Income' as ordinal
maxstat_test(jobsatisfaction,
             scores = list("Job.Satisfaction" = 1:4,
                           "Income" = 1:4))
```

`mercuryfish` *Chromosomal Effects of Mercury-Contaminated Fish Consumption*

Description

The mercury level in blood, the proportion of cells with abnormalities, and the proportion of cells with chromosome aberrations in consumers of mercury-contaminated fish and a control group.

Usage

```
mercuryfish
```

Format

A data frame with 39 observations on 4 variables.

`group` a factor with levels `"control"` and `"exposed"`.

`mercury` mercury level in blood.

`abnormal` the proportion of cells with structural abnormalities.

`ccells` the proportion of C_u cells, i.e., cells with asymmetrical or incomplete-symmetrical chromosome aberrations.

Details

Control subjects (`"control"`) and subjects who ate contaminated fish for more than three years (`"exposed"`) are under study (Skerfving, Hansson, Mangs, Lindsten, and Ryman 1974).

Rosenbaum (1994) proposed a coherence criterion defining a partial ordering, i.e., an observation is smaller than another when all responses are smaller, and a score reflecting the "ranking" is attached to each observation. The corresponding partially ordered set (POSET) test can be used to test if the distribution of the scores differ between the groups. Alternatively, a multivariate test can be applied (Hothorn, Hornik, van de Wiel, and Zeileis 2006).

References

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2006). "A Lego System for Conditional Inference." *The American Statistician*, **60**(3), 257–263. doi:10.1198/000313006X118430.

Rosenbaum PR (1994). "Coherence in Observational Studies." *Biometrics*, **50**(2), 368–374. doi:10.2307/2533380.

Skerfving S, Hansson K, Mangs C, Lindsten J, Ryman N (1974). "Methylmercury-induced Chromosome Damage in Man." *Environmental Research*, **7**(1), 83–98. doi:10.1016/00139351(74)90078-4.

Examples

```
## Coherence criterion
coherence <- function(data) {
    x <- as.matrix(data)
    matrix(apply(x, 1, function(y)
        sum(colSums(t(x) < y) == ncol(x)) -
            sum(colSums(t(x) > y) == ncol(x))), ncol = 1)
}

## Asymptotic POSET test
poset <- independence_test(mercury + abnormal + ccells ~ group,
                           data = mercuryfish, ytrafo = coherence)

## Linear statistic (T in the notation of Rosenbaum, 1994)
statistic(poset, type = "linear")

## Expectation
expectation(poset)

## Variance
## Note: typo in Rosenbaum (1994, p. 371, Sec. 2, last paragraph)
variance(poset)

## Standardized statistic
statistic(poset)

## P-value
pvalue(poset)

## Exact POSET test
independence_test(mercury + abnormal + ccells ~ group,
                  data = mercuryfish, ytrafo = coherence,
                  distribution = "exact")

## Asymptotic multivariate test
mvtest <- independence_test(mercury + abnormal + ccells ~ group,
                            data = mercuryfish)

## Global p-value
pvalue(mvtest)

## Single-step adjusted p-values
pvalue(mvtest, method = "single-step")

## Step-down adjusted p-values
pvalue(mvtest, method = "step-down")
```

---

neuropathy *Acute Painful Diabetic Neuropathy*

Description

The logarithm of the ratio of pain scores measured at baseline and after four weeks in a control group and a treatment group.

Usage

```
neuropathy
```

Format

A data frame with 58 observations on 2 variables.

`pain` pain scores: ln(baseline / final).

`group` a factor with levels `"control"` and `"treat"`.

Details

Data from Conover and Salsburg (1988, Tab. 1).

Source

Conover WJ, Salsburg DS (1988). "Locally Most Powerful Tests for Detecting Treatment Effects When Only a Subset of Patients Can Be Expected to "Respond" to Treatment." *Biometrics*, **44**(1), 189. ISSN 0006-341X. doi:10.2307/2531906.

Examples

```
## Conover and Salsburg (1988, Tab. 2)

## One-sided approximative Fisher-Pitman test
oneway_test(pain ~ group, data = neuropathy,
            alternative = "less",
            distribution = approximate(nresample = 10000))

## One-sided approximative Wilcoxon-Mann-Whitney test
wilcox_test(pain ~ group, data = neuropathy,
            alternative = "less",
            distribution = approximate(nresample = 10000))

## One-sided approximative Conover-Salsburg test
oneway_test(pain ~ group, data = neuropathy,
            alternative = "less",
            distribution = approximate(nresample = 10000),
            ytrafo = function(data)
                trafo(data, numeric_trafo = consal_trafo))

## One-sided approximative maximum test for a range of 'a' values
it <- independence_test(pain ~ group, data = neuropathy,
                        alternative = "less",
                        distribution = approximate(nresample = 10000),
                        ytrafo = function(data)
```

```
                            trafo(data, numeric_trafo = function(y)
                                consal_trafo(y, a = 2:7)))
pvalue(it, method = "single-step")
```

`NullDistribution` *Specification of the Reference Distribution*

Description

Specification of the asymptotic, approximative (Monte Carlo) and exact reference distribution.

Usage

```
asymptotic(maxpts = 25000, abseps = 0.001, releps = 0)
approximate(nresample = 10000L, parallel = c("no", "multicore", "snow"),
            ncpus = 1L, cl = NULL, B)
exact(algorithm = c("auto", "shift", "split-up"), fact = NULL)
```

Arguments

`maxpts`	an integer, the maximum number of function values. Defaults to `25000`.
`abseps`	a numeric, the absolute error tolerance. Defaults to `0.001`.
`releps`	a numeric, the relative error tolerance. Defaults to `0`.
`nresample`	a positive integer, the number of Monte Carlo replicates used for the computation of the approximative reference distribution. Defaults to `10000L`.
`parallel`	a character, the type of parallel operation: either `"no"` (default), `"multicore"` or `"snow"`.
`ncpus`	an integer, the number of processes to be used in parallel operation. Defaults to `1L`.
`cl`	an object inheriting from class `"cluster"`, specifying an optional **parallel** or **snow** cluster if `parallel = "snow"`. Defaults to `NULL`.
`B`	deprecated, use `nresample` instead.
`algorithm`	a character, the algorithm used for the computation of the exact reference distribution: either `"auto"` (default), `"shift"` or `"split-up"`.
`fact`	an integer to multiply the response values with. Defaults to `NULL`.

Details

`asymptotic()`, `approximate()` and `exact()` can be supplied to the `distribution` argument of, e.g., `independence_test()` to provide control of the specification of the asymptotic, approximative (Monte Carlo) and exact reference distribution, respectively.

The asymptotic reference distribution is computed using a randomised quasi-Monte Carlo method (Genz and Bretz 2009) and is applicable to arbitrary covariance structures with dimensions up to 1000. See `GenzBretz()` in package **mvtnorm** for details on `maxpts`, `abseps` and `releps`.

The approximative (Monte Carlo) reference distribution is obtained by a conditional Monte Carlo procedure, i.e., by computing the test statistic for `nresample` random samples from all admissible permutations of the response $\mathbf{Y}$ within each block (Hothorn, Hornik, van de Wiel, and Zeileis 2008). By default, the distribution is computed using serial operation (`parallel = "no"`). The use of parallel operation is specified by setting `parallel` to either `"multicore"` (not available for MS Windows) or `"snow"`. In the latter case, if `cl = NULL` (default) a cluster with `ncpus` processes is created on the local machine unless a default cluster has been registered (see `setDefaultCluster()` in package **parallel**) in which case that gets used instead. Alternatively, the use of an optional **parallel** or **snow** cluster can be specified by `cl`. See 'Examples' and package **parallel** for details on parallel operation.

The exact reference distribution, currently available for univariate two-sample problems only, is computed using either the shift algorithm (Streitberg and Röhmel 1984; Streitberg and Röhmel 1986; Streitberg and Röhmel 1987) or the split-up algorithm (van de Wiel 2001). The shift algorithm handles blocks pertaining to, e.g., pre- and post-stratification, but can only be used with positive integer-valued scores $h(\mathbf{Y})$. The split-up algorithm can be used with non-integer scores, but does not handle blocks. By default, an automatic choice is made (`algorithm = "auto"`) but the shift and split-up algorithms can be selected by setting `algorithm` to `"shift"` or `"split-up"`, respectively.

Note

Starting with version 1.1-0, the default for `algorithm` is `"auto"`, having identical behaviour to `"shift"` in previous versions. In earlier versions of the package, `algorithm = "shift"` silently switched to the split-up algorithm if non-integer scores were detected, whereas the current version exits with a warning.

In versions prior to 1.3-0, the number of Monte Carlo replicates in `approximate()` was specified using the now deprecated `B` argument. **This will be made defunct and removed in a future release.** It has been replaced by the `nresample` argument (for consistency with the **libcoin**, **party** and **partykit** packages).

References

Genz A, Bretz F (2009). *Computation of Multivariate Normal and t Probabilities*, series Lecture Notes in Statistics. Springer-Verlag, Heidelberg, Germany. ISBN 978-3-642-01688-2.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2008). "Implementing a Class of Permutation Tests: The **coin** Package." *Journal of Statistical Software*, **28**(8), 1–23. doi:10.18637/jss.v028.i08.

Streitberg B, Röhmel J (1984). "Exact Nonparametrics in APL." *ACM SIGAPL APL Quote Quad*, **14**(4), 313–325. doi:10.1145/384283.801115.

Streitberg B, Röhmel J (1986). "Exact Distributions for Permutation and Rank Tests: An Introduction to Some Recently Published Algorithms." *Statistical Software Newsletter*, **12**(1), 10–17. ISSN 1609-3631.

Streitberg B, Röhmel J (1987). "Exakte Verteilungen für Rang- und Randomisierungstests im allgemeinen c-Stichprobenfall." *EDV in Medizin und Biologie*, **18**(1), 12–19.

van de Wiel MA (2001). "The Split-Up Algorithm: A Fast Symbolic Method for Computing p-Values of Distribution-Free Statistics." *Computational Statistics*, **16**, 519–538.

Examples

```
## Approximative (Monte Carlo) Cochran-Mantel-Haenszel test

## Serial operation
set.seed(123)
cmh_test(disease ~ smoking | gender, data = alzheimer,
         distribution = approximate(nresample = 100000))

## Not run:
## Multicore with 8 processes (not for MS Windows)
set.seed(123, kind = "L'Ecuyer-CMRG")
cmh_test(disease ~ smoking | gender, data = alzheimer,
         distribution = approximate(nresample = 100000,
                                    parallel = "multicore", ncpus = 8))

## Automatic PSOCK cluster with 4 processes
set.seed(123, kind = "L'Ecuyer-CMRG")
cmh_test(disease ~ smoking | gender, data = alzheimer,
         distribution = approximate(nresample = 100000,
                                    parallel = "snow", ncpus = 4))

## Registered FORK cluster with 12 processes (not for MS Windows)
fork12 <- parallel::makeCluster(12, "FORK") # set-up cluster
parallel::setDefaultCluster(fork12) # register default cluster
set.seed(123, kind = "L'Ecuyer-CMRG")
cmh_test(disease ~ smoking | gender, data = alzheimer,
         distribution = approximate(nresample = 100000,
                                    parallel = "snow"))
parallel::stopCluster(fork12) # clean-up

## User-specified PSOCK cluster with 8 processes
psock8 <- parallel::makeCluster(8, "PSOCK") # set-up cluster
set.seed(123, kind = "L'Ecuyer-CMRG")
cmh_test(disease ~ smoking | gender, data = alzheimer,
         distribution = approximate(nresample = 100000,
                                    parallel = "snow", cl = psock8))
parallel::stopCluster(psock8) # clean-up
## End(Not run)
```

NullDistribution-class

Class "NullDistribution" *and Its Subclasses*

Description

Objects of class "NullDistribution" and its subclasses "ApproxNullDistribution", "AsymptNullDistribution" and "ExactNullDistribution" represent the reference distribution.

Objects from the Class

Objects can be created by calls of the form

```
new("NullDistribution", ...),

new("ApproxNullDistribution", ...),

new("AsymptNullDistribution", ...)
```

and

```
new("ExactNullDistribution", ...).
```

Slots

For objects of classes "NullDistribution", "ApproxNullDistribution", "AsymptNullDistribution" or "ExactNullDistribution":

name: Object of class "character". The name of the reference distribution.

p: Object of class "function". The distribution function of the reference distribution.

pvalue: Object of class "function". The p-value function of the reference distribution.

parameters: Object of class "list". Additional parameters.

support: Object of class "function". The support of the reference distribution.

d: Object of class "function". The density function of the reference distribution.

q: Object of class "function". The quantile function of the reference distribution.

midpvalue: Object of class "function". The mid-p-value function of the reference distribution.

pvalueinterval: Object of class "function". The p-value interval function of the reference distribution.

size: Object of class "function". The size function of the reference distribution.

Additionally, for objects of classes "ApproxNullDistribution" or "AsymptNullDistribution":

seed: Object of class "integer". The random number generator state (i.e., the value of .Random.seed).

Additionally, for objects of class "ApproxNullDistribution":

nresample: Object of class "numeric". The number of Monte Carlo replicates.

Extends

For objects of class "NullDistribution":
Class "PValue", directly.

For objects of classes "ApproxNullDistribution", "AsymptNullDistribution" or "ExactNullDistribution":
Class "NullDistribution", directly.
Class "PValue", by class "NullDistribution", distance 2.

Known Subclasses

For objects of class "NullDistribution":
Class "ApproxNullDistribution", directly.
Class "AsymptNullDistribution", directly.
Class "ExactNullDistribution", directly.

Methods

dperm signature(object = "NullDistribution"): See the documentation for dperm() for details.

midpvalue signature(object = "NullDistribution"): See the documentation for midpvalue() for details.

midpvalue signature(object = "ApproxNullDistribution"): See the documentation for midpvalue() for details.

pperm signature(object = "NullDistribution"): See the documentation for pperm() for details.

pvalue signature(object = "NullDistribution"): See the documentation for pvalue() for details.

pvalue signature(object = "ApproxNullDistribution"): See the documentation for pvalue() for details.

pvalue_interval signature(object = "NullDistribution"): See the documentation for pvalue_interval() for details.

qperm signature(object = "NullDistribution"): See the documentation for qperm() for details.

rperm signature(object = "NullDistribution"): See the documentation for rperm() for details.

size signature(object = "NullDistribution"): See the documentation for size() for details.

support signature(object = "NullDistribution"): See the documentation for support() for details.

NullDistribution-methods

Computation of the Reference Distribution

Description

Methods for computation of the asymptotic, approximative (Monte Carlo) and exact reference distribution.

Usage

```
## S4 method for signature 'MaxTypeIndependenceTestStatistic'
AsymptNullDistribution(object, ...)
## S4 method for signature 'QuadTypeIndependenceTestStatistic'
AsymptNullDistribution(object, ...)
## S4 method for signature 'ScalarIndependenceTestStatistic'
AsymptNullDistribution(object, ...)

## S4 method for signature 'MaxTypeIndependenceTestStatistic'
ApproxNullDistribution(object, nresample = 10000L, B, ...)
## S4 method for signature 'QuadTypeIndependenceTestStatistic'
ApproxNullDistribution(object, nresample = 10000L, B, ...)
## S4 method for signature 'ScalarIndependenceTestStatistic'
ApproxNullDistribution(object, nresample = 10000L, B, ...)

## S4 method for signature 'QuadTypeIndependenceTestStatistic'
ExactNullDistribution(object, algorithm = c("auto", "shift", "split-up"), ...)
## S4 method for signature 'ScalarIndependenceTestStatistic'
ExactNullDistribution(object, algorithm = c("auto", "shift", "split-up"), ...)
```

Arguments

`object`	an object from which the asymptotic, approximative (Monte Carlo) or exact reference distribution can be computed.
`nresample`	a positive integer, the number of Monte Carlo replicates used for the computation of the approximative reference distribution. Defaults to `10000L`.
`B`	deprecated, use `nresample` instead.
`algorithm`	a character, the algorithm used for the computation of the exact reference distribution: either `"auto"` (default), `"shift"` or `"split-up"`.
`...`	further arguments to be passed to or from methods.

Details

The methods `AsymptNullDistribution`, `ApproxNullDistribution` and `ExactNullDistribution` compute the asymptotic, approximative (Monte Carlo) and exact reference distribution, respectively.

Value

An object of class `"AsymptNullDistribution"`, `"ApproxNullDistribution"` or `"ExactNullDistribution"`.

Note

In versions prior to 1.3-0, the number of Monte Carlo replicates in `ApproxNullDistribution()` was specified using the now deprecated `B` argument. **This will be made defunct and removed in a future release.** It has been replaced by the `nresample` argument (for consistency with the **libcoin**, **party** and **partykit** packages).

`ocarcinoma` *Ovarian Carcinoma*

Description

Survival times of 35 women suffering from ovarian carcinoma at stadium II and IIA.

Usage

```
ocarcinoma
```

Format

A data frame with 35 observations on 3 variables.

`time` time (days).

`stadium` a factor with levels `"II"` and `"IIA"`.

`event` status indicator for `time`: `FALSE` for right-censored observations and `TRUE` otherwise.

Details

Data from Fleming, Green, and Harrington (1984) and Fleming, O'Fallon, O'Brien, and Harrington (1980) were reanalysed by Schumacher and Schulgen (2002).

References

Fleming TR, Green SJ, Harrington DP (1984). "Considerations for Monitoring and Evaluating Treatment Effects in Clinical Trials." *Controlled Clinical Trials*, **5**(1), 55–66. doi:10.1016/0197-2456(84)901508.

Fleming TR, O'Fallon JR, O'Brien PC, Harrington DP (1980). "Modified Kolmogorov-Smirnov Test Procedures with Application to Arbitrarily Right-Censored Data." *Biometrics*, **36**(4), 607–625. doi:10.2307/2556114.

Schumacher M, Schulgen G (2002). *Methodik Klinischer Studien*. Springer-Verlag, Heidelberg, Germany.

Examples

```
## Exact logrank test
lt <- logrank_test(Surv(time, event) ~ stadium, data = ocarcinoma,
                   distribution = "exact")

## Test statistic
statistic(lt)

## P-value
pvalue(lt)
```

PermutationDistribution-methods
Computation of the Permutation Distribution

Description

Methods for computation of the density function, distribution function, quantile function, random numbers and support of the permutation distribution.

Usage

```
## S4 method for signature 'NullDistribution'
dperm(object, x, ...)
## S4 method for signature 'IndependenceTest'
dperm(object, x, ...)

## S4 method for signature 'NullDistribution'
pperm(object, q, ...)
## S4 method for signature 'IndependenceTest'
pperm(object, q, ...)

## S4 method for signature 'NullDistribution'
qperm(object, p, ...)
## S4 method for signature 'IndependenceTest'
qperm(object, p, ...)

## S4 method for signature 'NullDistribution'
rperm(object, n, ...)
## S4 method for signature 'IndependenceTest'
rperm(object, n, ...)

## S4 method for signature 'NullDistribution'
support(object, ...)
## S4 method for signature 'IndependenceTest'
support(object, ...)
```

Arguments

object	an object from which the density function, distribution function, quantile function, random numbers or support of the permutation distribution can be computed.
x, q	a numeric vector, the quantiles for which the density function or distribution function is computed.
p	a numeric vector, the probabilities for which the quantile function is computed.
n	a numeric vector, the number of observations. If `length(n) > 1`, the length is taken to be the number required.
...	further arguments to be passed to methods.

Details

The methods `dperm`, `pperm`, `qperm`, `rperm` and `support` compute the density function, distribution function, quantile function, random deviates and support, respectively, of the permutation distribution.

Value

The density function, distribution function, quantile function, random deviates or support of the permutation distribution computed from `object`. A numeric vector.

Note

The density of asymptotic permutation distributions for maximum-type tests or exact permutation distributions obtained by the split-up algorithm is reported as `NA`. The quantile function of asymptotic permutation distributions for maximum-type tests cannot be computed for `p` less than 0.5, due to limitations in the **mvtnorm** package. The support of exact permutation distributions obtained by the split-up algorithm is reported as `NA`.

In versions prior to 1.1-0, the support of asymptotic permutation distributions was given as an interval containing 99.999 % of the probability mass. It is now reported as `NA`.

Examples

```
## Two-sample problem
dta <- data.frame(
    y = rnorm(20),
    x = gl(2, 10)
)

## Exact Ansari-Bradley test
at <- ansari_test(y ~ x, data = dta, distribution = "exact")

## Support of the exact distribution of the Ansari-Bradley statistic
supp <- support(at)

## Density of the exact distribution of the Ansari-Bradley statistic
dens <- dperm(at, x = supp)

## Plotting the density
plot(supp, dens, type = "s")

## 95% quantile
qperm(at, p = 0.95)

## One-sided p-value
pperm(at, q = statistic(at))

## Random number generation
rperm(at, n = 5)
```

photocar *Multiple Dosing Photococarcinogenicity Experiment*

Description

Survival time, time to first tumor, and total number of tumors in three groups of animals in a photococarcinogenicity study.

Usage

```
photocar
```

Format

A data frame with 108 observations on 6 variables.

`group` a factor with levels `"A"`, `"B"`, and `"C"`.

`ntumor` total number of tumors.

`time` survival time.

`event` status indicator for `time`: `FALSE` for right-censored observations and `TRUE` otherwise.

`dmin` time to first tumor.

`tumor` status indicator for `dmin`: `FALSE` when no tumor was observed and `TRUE` otherwise.

Details

The animals were exposed to different levels of ultraviolet radiation (UVR) exposure (group A: topical vehicle and 600 Robertson–Berger units of UVR, group B: no topical vehicle and 600 Robertson–Berger units of UVR and group C: no topical vehicle and 1200 Robertson–Berger units of UVR). The data are taken from Tables 1 to 3 in Molefe, Chen, Howard, Miller, Sambuco, Donald Forbes, and Kodell (2005).

The main interest is testing the global null hypothesis of no treatment effect with respect to survival time, time to first tumor and number of tumors. Molefe et al. (2005) also analyzed the detection time of tumors, but that data is not given here. In case the global null hypothesis can be rejected, the deviations from the partial null hypotheses are of special interest.

Source

Molefe DF, Chen JJ, Howard PC, Miller BJ, Sambuco CP, Donald Forbes P, Kodell RL (2005). "Tests for Effects on Tumor Frequency and Latency in Multiple Dosing Photococarcinogenicity Experiments." *Journal of Statistical Planning and Inference*, **129**(1–2), 39–58. doi:10.1016/j.jspi.2004.06.038.

Examples

```
## Plotting data
op <- par(no.readonly = TRUE) # save current settings
layout(matrix(1:3, ncol = 3))
with(photocar, {
    plot(survfit(Surv(time, event) ~ group),
         lty =  1:3, xmax = 50, main = "Survival Time")
    legend("bottomleft", lty = 1:3, levels(group), bty = "n")
    plot(survfit(Surv(dmin, tumor) ~ group),
         lty = 1:3, xmax = 50, main = "Time to First Tumor")
    legend("bottomleft", lty = 1:3, levels(group), bty = "n")
    boxplot(ntumor ~ group, main = "Number of Tumors")
})
par(op) # reset

## Approximative multivariate (all three responses) test
it <- independence_test(Surv(time, event) + Surv(dmin, tumor) + ntumor ~ group,
                        data = photocar,
                        distribution = approximate(nresample = 10000))

## Global p-value
pvalue(it)

## Why was the global null hypothesis rejected?
statistic(it, type = "standardized")
pvalue(it, method = "single-step")
```

PValue-class *Class* "PValue"

Description

Objects of class "PValue" represent the p-value, mid-p-value and p-value interval of the reference distribution.

Objects from the Class

Objects can be created by calls of the form

```
new("PValue", \dots).
```

Slots

name: Object of class "character". The name of the reference distribution.

p: Object of class "function". The distribution function of the reference distribution.

pvalue: Object of class "function". The p-value function of the reference distribution.

Methods

pvalue `signature(object = "PValue")`: See the documentation for `pvalue` for details.

Note

Starting with version 1.3-0, this class is deprecated and will be replaced by class `"NullDistribution"`. **It will be made defunct and removed in a future release.**

`pvalue-methods` *Computation of the p-Value, Mid-p-Value, p-Value Interval and Test Size*

Description

Methods for computation of the p-value, mid-p-value, p-value interval and test size.

Usage

```
## S4 method for signature 'PValue'
pvalue(object, q, ...)
## S4 method for signature 'NullDistribution'
pvalue(object, q, ...)
## S4 method for signature 'ApproxNullDistribution'
pvalue(object, q, ...)
## S4 method for signature 'IndependenceTest'
pvalue(object, ...)
## S4 method for signature 'MaxTypeIndependenceTest'
pvalue(object, method = c("global", "single-step",
                          "step-down", "unadjusted"),
       distribution = c("joint", "marginal"),
       type = c("Bonferroni", "Sidak"), ...)

## S4 method for signature 'NullDistribution'
midpvalue(object, q, ...)
## S4 method for signature 'ApproxNullDistribution'
midpvalue(object, q, ...)
## S4 method for signature 'IndependenceTest'
midpvalue(object, ...)

## S4 method for signature 'NullDistribution'
pvalue_interval(object, q, ...)
## S4 method for signature 'IndependenceTest'
pvalue_interval(object, ...)

## S4 method for signature 'NullDistribution'
size(object, alpha, type = c("p-value", "mid-p-value"), ...)
## S4 method for signature 'IndependenceTest'
size(object, alpha, type = c("p-value", "mid-p-value"), ...)
```

Arguments

object	an object from which the p-value, mid-p-value, p-value interval or test size can be computed.
q	a numeric, the quantile for which the p-value, mid-p-value or p-value interval is computed.
method	a character, the method used for the p-value computation: either "global" (default), "single-step", "step-down" or "unadjusted".
distribution	a character, the distribution used for the computation of adjusted p-values: either "joint" (default) or "marginal".
type	pvalue(): a character, the type of p-value adjustment when the marginal distributions are used: either "Bonferroni" (default) or "Sidak". size(): a character, the type of rejection region used when computing the test size: either "p-value" (default) or "mid-p-value".
alpha	a numeric, the nominal significance level α at which the test size is computed.
...	further arguments (currently ignored).

Details

The methods pvalue, midpvalue, pvalue_interval and size compute the p-value, mid-p-value, p-value interval and test size, respectively.

For pvalue(), the global p-value (method = "global") is returned by default and is given with an associated 99% confidence interval when resampling is used to determine the null distribution (which for maximum statistics may be true even in the asymptotic case).

The familywise error rate (FWER) is always controlled under the global null hypothesis, i.e., in the *weak* sense, implying that the smallest adjusted p-value is valid without further assumptions. Control of the FWER under any partial configuration of the null hypotheses, i.e., in the *strong* sense, as is typically desired for multiple tests and comparisons, requires that the *subset pivotality* condition holds, see Westfall and Young (1993, pp. 42–43) and Bretz, Hothorn, and Westfall (2010, pp. 136–137). In addition, for methods based on the joint distribution of the test statistics, failure of the *joint exchangeability* assumption, explained by Westfall and Troendle (2008) and in Bretz et al. (2010, pp. 129–130), may cause excess Type I errors.

Assuming *subset pivotality*, single-step or *free* step-down adjusted p-values using max-T procedures are obtained by setting method to "single-step" or "step-down", respectively. In both cases, the distribution argument specifies whether the adjustment is based on the joint distribution ("joint") or the marginal distributions ("marginal") of the test statistics. For procedures based on the marginal distributions, Bonferroni- or Šidák-type adjustment can be specified through the type argument by setting it to "Bonferroni" or "Sidak", respectively.

The p-value adjustment procedures based on the joint distribution of the test statistics fully utilizes distributional characteristics, such as discreteness and dependence structure, whereas procedures using the marginal distributions only incorporate discreteness. Hence, the joint distribution-based procedures are typically more powerful. Details regarding the single-step and *free* step-down procedures based on the joint distribution can be found in Westfall and Young (1993) in particular, this implementation uses Equation 2.8 with Algorithm 2.5 and 2.8, respectively. Westfall and Wolfinger (1997) provide details of the marginal distributions-based single-step and *free* step-down procedures. The generalization of Westfall and Wolfinger (1997) to arbitrary test statistics, as implemented here, is given Westfall and Troendle (2008).

Unadjusted p-values are obtained using `method = "unadjusted"`.

For `midpvalue()`, the global mid-p-value is given with an associated 99% mid-p confidence interval when resampling is used to determine the null distribution. The two-sided mid-p-value is computed according to the minimum likelihood method (Hirji, Tan, and Elashoff 1991).

The p-value interval $(p_0, p_1]$ obtained by `pvalue_interval()` was proposed by Berger (2000); Berger (2001), where the upper endpoint p_1 is the conventional p-value and the mid-point, i.e., $p_{0.5}$, is the mid-p-value. The lower endpoint p_0 is the smallest p-value attainable if no conservatism attributable to the discreteness of the null distribution is present. The length of the p-value interval is the null probability of the observed outcome and provides a data-dependent measure of conservatism that is completely independent of the nominal significance level.

For `size()`, the test size, i.e., the actual significance level, at the nominal significance level α is computed using either the rejection region corresponding to the p-value (`type = "p-value"`, default) or the mid-p-value (`type = "mid-p-value"`). The test size is, in contrast to the p-value interval, a data-independent measure of conservatism that depends on the nominal significance level. A test size smaller or larger than the nominal significance level indicates that the test procedure is conservative or anti-conservative, respectively, at that particular nominal significance level. However, as pointed out by Berger (2001), even when the actual and nominal significance levels are identical, conservatism may still affect the p-value.

Value

The p-value, mid-p-value, p-value interval or test size computed from `object`. A numeric vector or matrix.

Note

The mid-p-value, p-value interval and test size of asymptotic permutation distributions or exact permutation distributions obtained by the split-up algorithm is reported as `NA`.

In versions prior to 1.1-0, a min-P procedure computing Šidák single-step adjusted p-values accounting for discreteness was available when specifying `method = "discrete"`. This was made **defunct** in version 1.2-0 due to the introduction of a more general max-T version of the same algorithm.

References

Berger VW (2000). "Pros and Cons of Permutation Tests in Clinical Trials." *Statistics in Medicine*, **19**(10), 1319–1328.

Berger VW (2001). "The p-value Interval as an Inferential Tool." *Journal of the Royal Statistical Society: Series D (The Statistician)*, **50**(1), 79–85. doi:10.1111/14679884.00262.

Bretz F, Hothorn T, Westfall P (2010). *Multiple Comparisons Using R*. Chapman & Hall/CRC Press, Boca Raton, Florida, U.S.A. ISBN 978–1–58488–574–0.

Hirji KF, Tan S, Elashoff RM (1991). "A Quasi-exact Test for Comparing Two Binomial Proportions." *Statistics in Medicine*, **10**(7), 1137–1153. doi:10.1002/sim.4780100713.

Westfall PH, Troendle JF (2008). "Multiple Testing with Minimal Assumptions." *Biometrical Journal*, **50**(5), 745–755. doi:10.1002/bimj.200710456.

Westfall PH, Wolfinger RD (1997). "Multiple Tests with Discrete Distributions." *The American Statistician*, **51**(1), 3–8. doi:10.1080/00031305.1997.10473577.

Westfall PH, Young SS (1993). *Resampling-Based Multiple Testing*. John Wiley & Sons, New York, U.S.A.

Examples

```
## Two-sample problem
dta <- data.frame(
    y = rnorm(20),
    x = gl(2, 10)
)

## Exact Ansari-Bradley test
(at <- ansari_test(y ~ x, data = dta, distribution = "exact"))
pvalue(at)
midpvalue(at)
pvalue_interval(at)
size(at, alpha = 0.05)
size(at, alpha = 0.05, type = "mid-p-value")


## Bivariate two-sample problem
dta2 <- data.frame(
    y1 = rnorm(20) + rep(0:1, each = 10),
    y2 = rnorm(20),
    x = gl(2, 10)
)

## Approximative (Monte Carlo) bivariate Fisher-Pitman test
(it <- independence_test(y1 + y2 ~ x, data = dta2,
                         distribution = approximate(nresample = 10000)))

## Global p-value
pvalue(it)

## Joint distribution single-step p-values
pvalue(it, method = "single-step")

## Joint distribution step-down p-values
pvalue(it, method = "step-down")

## Sidak step-down p-values
pvalue(it, method = "step-down", distribution = "marginal", type = "Sidak")

## Unadjusted p-values
pvalue(it, method = "unadjusted")


## Length of YOY Gizzard Shad (Hollander and Wolfe, 1999, p. 200, Tab. 6.3)
yoy <- data.frame(
    length = c(46, 28, 46, 37, 32, 41, 42, 45, 38, 44,
               42, 60, 32, 42, 45, 58, 27, 51, 42, 52,
               38, 33, 26, 25, 28, 28, 26, 27, 27, 27,
               31, 30, 27, 29, 30, 25, 25, 24, 27, 30),
```

```
    site = gl(4, 10, labels = as.roman(1:4))
)

## Approximative (Monte Carlo) Fisher-Pitman test with contrasts
## Note: all pairwise comparisons
(it <- independence_test(length ~ site, data = yoy,
                         distribution = approximate(nresample = 10000),
                         xtrafo = mcp_trafo(site = "Tukey")))

## Joint distribution step-down p-values
pvalue(it, method = "step-down") # subset pivotality is violated
```

`rotarod` *Rotating Rats*

Description

The endurance time of 24 rats in two groups on a rotating cylinder.

Usage

```
rotarod
```

Format

A data frame with 24 observations on 2 variables.

`time` endurance time (seconds).

`group` a factor with levels `"control"` and `"treatment"`.

Details

The rats were randomly assigned to receive a fixed oral dose of a centrally acting muscle relaxant (`"treatment"`) or a saline solvent (`"control"`). The animals were placed on a rotating cylinder and the endurance time of each rat, i.e., the length of time each rat remained on the cylinder, was measured up to a maximum of 300 seconds.

This dataset is the basis of a comparison of 11 different software implementations of the Wilcoxon-Mann-Whitney test presented in Bergmann, Ludbrook, and Spooren (2000).

Note

The empirical variance in the control group is 0 and the group medians are identical. The exact conditional p-values are 0.0373 (two-sided) and 0.0186 (one-sided). The asymptotic two-sided p-value (corrected for ties) is 0.0147.

Source

Bergmann R, Ludbrook J, Spooren WPJM (2000). "Different Outcomes of the Wilcoxon-Mann-Whitney Test from Different Statistics Packages." *The American Statistician*, **54**(1), 72–77.

Examples

```
## One-sided exact Wilcoxon-Mann-Whitney test (p = 0.0186)
wilcox_test(time ~ group, data = rotarod, distribution = "exact",
            alternative = "greater")

## Two-sided exact Wilcoxon-Mann-Whitney test (p = 0.0373)
wilcox_test(time ~ group, data = rotarod, distribution = "exact")

## Two-sided asymptotic Wilcoxon-Mann-Whitney test (p = 0.0147)
wilcox_test(time ~ group, data = rotarod)
```

ScaleTests *Two- and K-Sample Scale Tests*

Description

Testing the equality of the distributions of a numeric response variable in two or more independent groups against scale alternatives.

Usage

```
## S3 method for class 'formula'
taha_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
taha_test(object, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
klotz_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
klotz_test(object, ties.method = c("mid-ranks", "average-scores"),
           conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
mood_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
mood_test(object, ties.method = c("mid-ranks", "average-scores"),
          conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
ansari_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
ansari_test(object, ties.method = c("mid-ranks", "average-scores"),
            conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
fligner_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
```

```
fligner_test(object, ties.method = c("mid-ranks", "average-scores"),
             conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula'
conover_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
conover_test(object, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

formula	a formula of the form y ~ x | block where y is a numeric variable, x is a factor and block is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations.
object	an object inheriting from class "IndependenceProblem".
conf.int	a logical indicating whether a confidence interval for the ratio of scales should be computed. Defaults to FALSE.
conf.level	a numeric, confidence level of the interval. Defaults to 0.95.
ties.method	a character, the method used to handle ties: the score generating function either uses mid-ranks ("mid-ranks", default) or averages the scores of randomly broken ties ("average-scores").
...	further arguments to be passed to independence_test().

Details

taha_test(), klotz_test(), mood_test(), ansari_test(), fligner_test() and conover_test() provide the Taha test, the Klotz test, the Mood test, the Ansari-Bradley test, the Fligner-Killeen test and the Conover-Iman test. A general description of these methods is given by Hollander and Wolfe (1999).For the adjustment of scores for tied values see Hájek, Šidák, and Sen (1999, pp. 133–135).

The null hypothesis of equality, or conditional equality given block, of the distribution of y in the groups defined by x is tested against scale alternatives. In the two-sample case, the two-sided null hypothesis is $H_0 : V(Y_1)/V(Y_2) = 1$, where $V(Y_s)$ is the variance of the responses in the sth sample. In case alternative = "less", the null hypothesis is $H_0 : V(Y_1)/V(Y_2) \geq 1$. When alternative = "greater", the null hypothesis is $H_0 : V(Y_1)/V(Y_2) \leq 1$. Confidence intervals for the ratio of scales are available and computed according to Bauer (1972).

The Fligner-Killeen test uses median centering in each of the samples, as suggested by Conover, Johnson, and Johnson (1981), whereas the Conover-Iman test, following Conover and Iman (1978), uses mean centering in each of the samples.

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (distribution = "asymptotic"). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for

univariate two-sample problems by setting `distribution` to `"approximate"` or `"exact"`, respectively. See `asymptotic()`, `approximate()` and `exact()` for details.

The example section uses data from Hollander and Wolfe (1999).

Value

An object inheriting from class `"IndependenceTest"`. Confidence intervals can be extracted by `confint()`.

Note

In the two-sample case, a *large* value of the Ansari-Bradley statistic indicates that sample 1 is *less* variable than sample 2, whereas a *large* value of the statistics due to Taha, Klotz, Mood, Fligner-Killeen, and Conover-Iman indicate that sample 1 is *more* variable than sample 2.

References

Bauer DF (1972). "Constructing Confidence Sets Using Rank Statistics." *Journal of the American Statistical Association*, 687–690. doi:10.1080/01621459.1972.10481279.

Conover WJ, Iman RL (1978). "Some Exact Tables for the Squared Ranks Test." *Communications in Statistics - Simulation and Computation*, **7**(5), 491–513. doi:10.1080/03610917808812093.

Conover WJ, Johnson ME, Johnson MM (1981). "A Comparative Study of Tests for Homogeneity of Variances, with Applications to the Outer Continental Shelf Bidding Data." *Technometrics*, **23**(4), 351–361. doi:10.1080/00401706.1981.10487680.

Hájek J, Šidák Z, Sen PK (1999). *Theory of Rank Tests*, 2nd edition. Academic Press, London, UK.

Hollander M, Wolfe DA (1999). *Nonparametric Statistical Inference*, 2nd edition. John Wiley & Sons, New York, U.S.A.

Examples

```
## Serum Iron Determination Using Hyland Control Sera
## Hollander and Wolfe (1999, p. 147, Tab 5.1)
sid <- data.frame(
    serum = c(111, 107, 100, 99, 102, 106, 109, 108, 104, 99,
              101, 96, 97, 102, 107, 113, 116, 113, 110, 98,
              107, 108, 106, 98, 105, 103, 110, 105, 104,
              100, 96, 108, 103, 104, 114, 114, 113, 108, 106, 99),
    method = gl(2, 20, labels = c("Ramsay", "Jung-Parekh"))
)

## Asymptotic Ansari-Bradley test
ansari_test(serum ~ method, data = sid)

## Exact Ansari-Bradley test
pvalue(ansari_test(serum ~ method, data = sid,
                   distribution = "exact"))


## Platelet Counts of Newborn Infants
```

```
## Hollander and Wolfe (1999, p. 171, Tab. 5.4)
platelet <- data.frame(
    counts = c(120, 124, 215, 90, 67, 95, 190, 180, 135, 399,
               12, 20, 112, 32, 60, 40),
    treatment = factor(rep(c("Prednisone", "Control"), c(10, 6)))
)

## Approximative (Monte Carlo) Lepage test
## Hollander and Wolfe (1999, p. 172)
lepage_trafo <- function(y)
    cbind("Location" = rank_trafo(y), "Scale" = ansari_trafo(y))

independence_test(counts ~ treatment, data = platelet,
                  distribution = approximate(nresample = 10000),
                  ytrafo = function(data)
                      trafo(data, numeric_trafo = lepage_trafo),
                  teststat = "quadratic")

## Why was the null hypothesis rejected?
## Note: maximum statistic instead of quadratic form
ltm <- independence_test(counts ~ treatment, data = platelet,
                         distribution = approximate(nresample = 10000),
                         ytrafo = function(data)
                             trafo(data, numeric_trafo = lepage_trafo))

## Step-down adjustment suggests a difference in location
pvalue(ltm, method = "step-down")

## The same results are obtained from the simple Sidak-Holm procedure since the
## correlation between Wilcoxon and Ansari-Bradley test statistics is zero
cov2cor(covariance(ltm))
pvalue(ltm, method = "step-down", distribution = "marginal", type = "Sidak")
```

`statistic-methods` *Extraction of the Test Statistic and the Linear Statistic*

Description

Methods for extraction of the test statistic and the linear statistic.

Usage

```
## S4 method for signature 'IndependenceLinearStatistic'
statistic(object, type = c("test", "linear", "centered", "standardized"),
          partial = FALSE, ...)
## S4 method for signature 'IndependenceTestStatistic'
statistic(object, type = c("test", "linear", "centered", "standardized"),
          partial = FALSE, ...)
## S4 method for signature 'IndependenceTest'
statistic(object, type = c("test", "linear", "centered", "standardized"),
          partial = FALSE, ...)
```

Arguments

object	an object from which the test statistic or the linear statistic can be extracted.
type	a character string indicating the type of statistic: either "test" (default) for the test statistic, "linear" for the unstandardized linear statistic, "centered" for the centered linear statistic or "standardized" for the standardized linear statistic.
partial	a logical indicating that the partial linear statistic for each block should be extracted. Defaults to FALSE.
...	further arguments (currently ignored).

Details

The method statistic extracts the univariate test statistic or the, possibly multivariate, linear statistic in its unstandardized, centered or standardized form.

The test statistic (type = "test") is returned by default. The unstandardized, centered or standardized linear statistic is obtained by setting type to "linear", "centered" or "standardized", respectively. For tests of conditional independence within blocks, the partial linear statistic for each block is obtained by setting partial = TRUE.

Value

The test statistic or the unstandardized, centered or standardized linear statistic extracted from object. A numeric vector, matrix or array.

Examples

```
## Example data
dta <- data.frame(
    y = gl(4, 5),
    x = gl(5, 4)
)

## Asymptotic Cochran-Mantel-Haenszel Test
ct <- cmh_test(y ~ x, data = dta)

## Test statistic
statistic(ct)

## The unstandardized linear statistic...
statistic(ct, type = "linear")

## ...is identical to the contingency table
xtabs(~ x + y, data = dta)

## The centered linear statistic...
statistic(ct, type = "centered")

## ...is identical to
statistic(ct, type = "linear") - expectation(ct)
```

```
## The standardized linear statistic, illustrating departures from the null
## hypothesis of independence...
statistic(ct, type = "standardized")

## ...is identical to
(statistic(ct, type = "linear") - expectation(ct)) / sqrt(variance(ct))
```

SurvivalTests *Two- and K-Sample Tests for Censored Data*

Description

Testing the equality of the survival distributions in two or more independent groups.

Usage

```
## S3 method for class 'formula'
logrank_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem'
logrank_test(object, ties.method = c("mid-ranks", "Hothorn-Lausen",
                                     "average-scores"),
             type = c("logrank", "Gehan-Breslow", "Tarone-Ware",
                      "Peto-Peto", "Prentice", "Prentice-Marek",
                      "Andersen-Borgan-Gill-Keiding",
                      "Fleming-Harrington", "Gaugler-Kim-Liao", "Self"),
             rho = NULL, gamma = NULL, ...)
```

Arguments

formula	a formula of the form `y ~ x | block` where `y` is a survival object (see `Surv` in package **survival**), `x` is a factor and `block` is an optional factor for stratification.
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to `NULL`.
weights	an optional formula of the form `~ w` defining integer valued case weights for each observation. Defaults to `NULL`, implying equal weight for all observations.
object	an object inheriting from class `"IndependenceProblem"`.
ties.method	a character, the method used to handle ties: the score generating function either uses mid-ranks (`"mid-ranks"`, default), the Hothorn-Lausen method (`"Hothorn-Lausen"`) or averages the scores of randomly broken ties (`"average-scores"`); see 'Details'.
type	a character, the type of test: either `"logrank"` (default), `"Gehan-Breslow"`, `"Tarone-Ware"`, `"Peto-Peto"`, `"Prentice"`, `"Prentice-Marek"`, `"Andersen-Borgan-Gill-Keiding"` `"Fleming-Harrington"`, `"Gaugler-Kim-Liao"` or `"Self"`; see 'Details'.

rho	a numeric, the ρ constant when type is "Tarone-Ware", "Fleming-Harrington", "Gaugler-Kim-Liao" or "Self"; see 'Details'. Defaults to NULL, implying 0.5 for type = "Tarone-Ware" and 0 otherwise.
gamma	a numeric, the γ constant when type is "Fleming-Harrington", "Gaugler-Kim-Liao" or "Self"; see 'Details'. Defaults to NULL, implying 0.
...	further arguments to be passed to independence_test().

Details

logrank_test() provides the weighted logrank test reformulated as a linear rank test. The family of weighted logrank tests encompasses a large collection of tests commonly used in the analysis of survival data including, but not limited to, the standard (unweighted) logrank test, the Gehan-Breslow test, the Tarone-Ware class of tests, the Peto-Peto test, the Prentice test, the Prentice-Marek test, the Andersen-Borgan-Gill-Keiding test, the Fleming-Harrington class of tests, the Gaugler-Kim-Liao class of tests and the Self class of tests. A general description of these methods is given by Klein and Moeschberger (2003, Ch. 7). See Letón and Zuluaga (2001) for the linear rank test formulation.

The null hypothesis of equality, or conditional equality given block, of the survival distribution of y in the groups defined by x is tested. In the two-sample case, the two-sided null hypothesis is $H_0 : \theta = 1$, where $\theta = \lambda_2/\lambda_1$ and λ_s is the hazard rate in the sth sample. In case alternative = "less", the null hypothesis is $H_0 : \theta \geq 1$, i.e., the survival is lower in sample 1 than in sample 2. When alternative = "greater", the null hypothesis is $H_0 : \theta \leq 1$, i.e., the survival is higher in sample 1 than in sample 2.

If x is an ordered factor, the default scores, 1:nlevels(x), can be altered using the scores argument (see independence_test()); this argument can also be used to coerce nominal factors to class "ordered". In this case, a linear-by-linear association test is computed and the direction of the alternative hypothesis can be specified using the alternative argument. This type of extension of the standard logrank test was given by Tarone (1975) and later generalized to general weights by Tarone and Ware (1977).

Let (t_i, δ_i), $i = 1, 2, \ldots, n$, represent a right-censored random sample of size n, where t_i is the observed survival time and δ_i is the status indicator (δ_i is 0 for right-censored observations and 1 otherwise). To allow for ties in the data, let $t_{(1)} < t_{(2)} < \cdots < t_{(m)}$ represent the m, $m \leq n$, ordered distinct event times. At time $t_{(k)}$, $k = 1, 2, \ldots, m$, the number of events and the number of subjects at risk are given by $d_k = \sum_{i=1}^{n} I\big(t_i = t_{(k)} \,|\, \delta_i = 1\big)$ and $n_k = n - r_k$, respectively, where r_k depends on the ties handling method.

Three different methods of handling ties are available using ties.method: mid-ranks ("mid-ranks", default), the Hothorn-Lausen method ("Hothorn-Lausen") and average-scores ("average-scores"). The first and last method are discussed and contrasted by Callaert (2003), whereas the second method is defined in Hothorn and Lausen (2003). The mid-ranks method leads to

$$r_k = \sum_{i=1}^{n} I\big(t_i < t_{(k)}\big)$$

whereas the Hothorn-Lausen method uses

$$r_k = \sum_{i=1}^{n} I\big(t_i \leq t_{(k)}\big) - 1.$$

The scores assigned to right-censored and uncensored observations at the kth event time are given by

$$C_k = \sum_{j=1}^{k} w_j \frac{d_j}{n_j} \quad \text{and} \quad c_k = C_k - w_k,$$

respectively, where w is the logrank weight. For the average-scores method, used by, e.g., the software package StatXact, the d_k events observed at the kth event time are arbitrarily ordered by assigning them distinct values $t_{(k_l)}$, $l = 1, 2, \ldots, d_k$, infinitesimally to the left of $t_{(k)}$. Then scores C_{k_l} and c_{k_l} are computed as indicated above, effectively assuming that no event times are tied. The scores C_k and c_k are assigned the average of the scores C_{k_l} and c_{k_l}, respectively. It then follows that the score for the ith subject is

$$a_i = \begin{cases} C_{k'} & \text{if } \delta_i = 0 \\ c_{k'} & \text{otherwise} \end{cases}$$

where $k' = \max\{k : t_i \geq t_{(k)}\}$.

The `type` argument allows for a choice between some of the most well-known members of the family of weighted logrank tests, each corresponding to a particular weight function. The standard logrank test (`"logrank"`, default) was suggested by Mantel (1966), Peto and Peto (1972) and Cox (1972) and has $w_k = 1$. The Gehan-Breslow test (`"Gehan-Breslow"`) proposed by Gehan (1965) and later extended to K samples by Breslow (1970) is a generalization of the Wilcoxon rank-sum test, where $w_k = n_k$. The Tarone-Ware class of tests (`"Tarone-Ware"`) discussed by Tarone and Ware (1977) has $w_k = n_k^\rho$, where ρ is a constant; $\rho = 0.5$ (default) was suggested by Tarone and Ware (1977), but note that $\rho = 0$ and $\rho = 1$ lead to the standard logrank test and Gehan-Breslow test, respectively. The Peto-Peto test (`"Peto-Peto"`) suggested by Peto and Peto (1972) is another generalization of the Wilcoxon rank-sum test, where

$$w_k = \hat{S}_k = \prod_{j=0}^{k-1} \frac{n_j - d_j}{n_j}$$

is the *left-continuous* Kaplan-Meier estimator of the survival function, $n_0 \equiv n$ and $d_0 \equiv 0$. The Prentice test (`"Prentice"`) is also a generalization of the Wilcoxon rank-sum test proposed by Prentice (1978), where

$$w_k = \prod_{j=1}^{k} \frac{n_j}{n_j + d_j}.$$

The Prentice-Marek test (`"Prentice-Marek"`) is yet another generalization of the Wilcoxon rank-sum test discussed by Prentice and Marek (1979), with

$$w_k = \tilde{S}_k = \prod_{j=1}^{k} \frac{n_j + 1 - d_j}{n_j + 1}.$$

The Andersen-Borgan-Gill-Keiding test (`"Andersen-Borgan-Gill-Keiding"`) suggested by Andersen, Borgan, Gill, Keiding, and Borgan (1982) is a modified version of the Prentice-Marek test using

$$w_k = \frac{n_k}{n_k + 1} \prod_{j=0}^{k-1} \frac{n_j + 1 - d_j}{n_j + 1},$$

where, again, $n_0 \equiv n$ and $d_0 \equiv 0$. The Fleming-Harrington class of tests (`"Fleming-Harrington"`) proposed by Fleming and Harrington (1991) uses $w_k = \hat{S}_k^\rho (1 - \hat{S}_k)^\gamma$, where ρ and γ are constants; $\rho = 0$ and $\gamma = 0$ lead to the standard logrank test, while $\rho = 1$ and $\gamma = 0$ result in the Peto-Peto test. The Gaugler-Kim-Liao class of tests (`"Gaugler-Kim-Liao"`) discussed by Gaugler, Kim, and Liao (2007) is a modified version of the Fleming-Harrington class of tests, replacing $\hat{S}_k$ with $\tilde{S}_k$ so that $w_k = \tilde{S}_k^\rho (1 - \tilde{S}_k)^\gamma$, where ρ and γ are constants; $\rho = 0$ and $\gamma = 0$ lead to the standard logrank test, whereas $\rho = 1$ and $\gamma = 0$ result in the Prentice-Marek test. The Self class of tests (`"Self"`) suggested by Self (1991) has $w_k = v_k^\rho (1 - v_k)^\gamma$, where

$$v_k = \frac{1}{2} \frac{t_{(k-1)} + t_{(k)}}{t_{(m)}}, \quad t_{(0)} \equiv 0$$

is the standardized mid-point between the $(k-1)$th and the kth event time. (This is a slight generalization of Self's original proposal in order to allow for non-integer follow-up times.) Again, ρ and γ are constants and $\rho = 0$ and $\gamma = 0$ lead to the standard logrank test.

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (`distribution = "asymptotic"`). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for univariate two-sample problems by setting `distribution` to `"approximate"` or `"exact"`, respectively. See `asymptotic()`, `approximate()` and `exact()` for details.

Value

An object inheriting from class `"IndependenceTest"`.

Note

Peto and Peto (1972) proposed the test statistic implemented in `logrank_test()` and named it the *logrank test*. However, the Mantel-Cox test (Mantel 1966; Cox 1972), as implemented in `survdiff()` (in package **survival**), is also known as the logrank test. These tests are similar, but differ in the choice of probability model: the (Peto-Peto) logrank test uses the permutational variance, whereas the Mantel-Cox test is based on the hypergeometric variance.

Combining `independence_test()` or `symmetry_test()` with `logrank_trafo()` offers more flexibility than `logrank_test()` and allows for, among other things, maximum-type versatile test procedures (e.g. Lee 1996, see 'Examples') and user-supplied logrank weights (see `GTSG` for tests against Weibull-type or crossing-curve alternatives).

Starting with version 1.1-0, `logrank_test()` replaced `surv_test()` which was made **defunct** in version 1.2-0. Furthermore, `logrank_trafo()` is now an increasing function for all choices of `ties.method`, implying that the test statistic has the same sign irrespective of the ties handling method. Consequently, the sign of the test statistic will now be the opposite of what it was in earlier versions unless `ties.method = "average-scores"`. (In versions prior to 1.1-0, `logrank_trafo()` was a decreasing function when `ties.method` was other than `"average-scores"`.)

Starting with version 1.2-0, mid-ranks and the Hothorn-Lausen method can no longer be specified with `ties.method = "logrank"` and `ties-method = "HL"`, respectively.

References

Andersen PK, Borgan Ø, Gill R, Keiding N, Borgan O (1982). "Linear Nonparametric Tests for Comparison of Counting Processes, with Applications to Censored Survival Data." *International*

Statistical Review / Revue Internationale de Statistique, **50**(3), 219–258. doi:10.2307/1402489.

Breslow N (1970). "A Generalized Kruskal-Wallis Test for Comparing k Samples Subject to Unequal Patterns of Censorship." *Biometrika*, **57**(3), 579–594. doi:10.1093/biomet/57.3.579.

Callaert H (2003). "Comparing Statistical Software Packages: The Case of the Logrank Test in StatXact." *The American Statistician*, **57**(3), 214–217. doi:10.1198/0003130031900.

Cox DR (1972). "Regression Models and Life-Tables." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **34**(2), 187–202. doi:10.1111/j.25176161.1972.tb00899.x.

Fleming TR, Harrington DP (1991). *Counting Processes and Survival Analysis*. John Wiley & Sons, New York, U.S.A.

Gaugler T, Kim D, Liao S (2007). "Comparing Two Survival Time Distributions: An Investigation of Several Weight Functions for the Weighted Logrank Statistic." *Communications in Statistics - Simulation and Computation*, **36**(2), 423–435. doi:10.1080/03610910601161272.

Gehan EA (1965). "A Generalized Wilcoxon Test for Comparing Arbitrarily Singly-censored Samples." *Biometrika*, **52**(1–2), 203–224. doi:10.1093/biomet/52.12.203.

Hothorn T, Lausen B (2003). "On the Exact Distribution of Maximally Selected Rank Statistics." *Computational Statistics & Data Analysis*, **43**(2), 121–137.

Klein JP, Moeschberger MK (2003). *Survival Analysis. Techniques for Censored and Truncated Data.*, 2nd edition. Springer-Verlag, New York, U.S.A.

Lee JW (1996). "Some Versatile Tests Based on the Simultaneous Use of Weighted Log-Rank Statistics." *Biometrics*, **52**(2), 721–725. doi:10.2307/2532911.

Letón E, Zuluaga P (2001). "Equivalence between Score and Weighted Tests for Survival Curves." *Communications in Statistics - Theory and Methods*, **30**(4), 591–608. doi:10.1081/sta100002138.

Mantel N (1966). "Evaluation of Survival Data and Two New Rank Order Statistics Arising in its Consideration." *Cancer Chemotherapy Reports*, **50**(3), 163–170.

Peto R, Peto J (1972). "Asymptotically Efficient Rank Invariant Test Procedures." *Journal of the Royal Statistical Society. Series A (General)*, **135**(2), 185–207. doi:10.2307/2344317.

Prentice RL (1978). "Linear Rank Tests with Right Censored Data." *Biometrika*, **65**(1), 167–179. doi:10.1093/biomet/65.1.167.

Prentice RL, Marek P (1979). "A Qualitative Discrepancy between Censored Data Rank Tests." *Biometrics*, **35**(4), 861–867. doi:10.2307/2530120.

Self SG (1991). "An Adaptive Weighted Log-Rank Test with Application to Cancer Prevention and Screening Trials." *Biometrics*, **47**(3), 975–986. doi:10.2307/2532653.

Tarone RE (1975). "Tests for Trend in Life Table Analysis." *Biometrika*, **62**(3), 679–690. doi:10.1093/biomet/62.3.679.

Tarone RE, Ware J (1977). "On Distribution-free Tests for Equality of Survival Distributions." *Biometrika*, **64**(1), 156–160. doi:10.1093/biomet/64.1.156.

Examples

```
## Example data (Callaert, 2003, Tab. 1)
callaert <- data.frame(
    time = c(1, 1, 5, 6, 6, 6, 6, 2, 2, 2, 3, 4, 4, 5, 5),
    group = factor(rep(0:1, c(7, 8)))
)
```

```
## Logrank scores using mid-ranks (Callaert, 2003, Tab. 2)
with(callaert,
     logrank_trafo(Surv(time)))

## Asymptotic Mantel-Cox test (p = 0.0523)
survdiff(Surv(time) ~ group, data = callaert)

## Exact logrank test using mid-ranks (p = 0.0505)
logrank_test(Surv(time) ~ group, data = callaert, distribution = "exact")

## Exact logrank test using average-scores (p = 0.0468)
logrank_test(Surv(time) ~ group, data = callaert, distribution = "exact",
             ties.method = "average-scores")


## Lung cancer data (StatXact 9 manual, p. 213, Tab. 7.19)
lungcancer <- data.frame(
    time = c(257, 476, 355, 1779, 355,
             191, 563, 242, 285, 16, 16, 16, 257, 16),
    event = c(0, 0, 1, 1, 0,
              1, 1, 1, 1, 1, 1, 1, 1, 1),
    group = factor(rep(1:2, c(5, 9)),
                   labels = c("newdrug", "control"))
)

## Logrank scores using average-scores (StatXact 9 manual, p. 214)
with(lungcancer,
     logrank_trafo(Surv(time, event), ties.method = "average-scores"))

## Exact logrank test using average-scores (StatXact 9 manual, p. 215)
logrank_test(Surv(time, event) ~ group, data = lungcancer,
             distribution = "exact", ties.method = "average-scores")

## Exact Prentice test using average-scores (StatXact 9 manual, p. 222)
logrank_test(Surv(time, event) ~ group, data = lungcancer,
             distribution = "exact", ties.method = "average-scores",
             type = "Prentice")


## Approximative (Monte Carlo) versatile test (Lee, 1996)
rho.gamma <- expand.grid(rho = seq(0, 2, 1), gamma = seq(0, 2, 1))
lee_trafo <- function(y)
    logrank_trafo(y, ties.method = "average-scores",
                  type = "Fleming-Harrington",
                  rho = rho.gamma["rho"], gamma = rho.gamma["gamma"])

it <- independence_test(Surv(time, event) ~ group, data = lungcancer,
                        distribution = approximate(nresample = 10000),
                        ytrafo = function(data)
                            trafo(data, surv_trafo = lee_trafo))
pvalue(it, method = "step-down")
```

SymmetryProblem-class *Class* "SymmetryProblem"

Description

Objects of class "SymmetryProblem" represent the data structure corresponding to a symmetry problem.

Objects from the Class

Objects can be created by calls of the form

```
new("SymmetryProblem", x, y, block = NULL, weights = NULL, ...)
```

where x and y are data frames containing the variables $\mathbf{X}$ and $\mathbf{Y}$, respectively, block is an optional factor representing the block structure b and weights is an optional integer vector corresponding to the case weights w.

Slots

x: Object of class "data.frame". The variables x.

y: Object of class "data.frame". The variables y.

block: Object of class "factor". The block structure.

weights: Object of class "numeric". The case weights. (Not yet implemented!)

Extends

Class "IndependenceProblem", directly.

Methods

initialize signature(.Object = "SymmetryProblem"): See the documentation for initialize() (in package **methods**) for details.

SymmetryTest *General Symmetry Test*

Description

Testing the symmetry of a set of variables measured on arbitrary scales in a complete block design.

Usage

```
## S3 method for class 'formula'
symmetry_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table'
symmetry_test(object, ...)
## S3 method for class 'SymmetryProblem'
symmetry_test(object, teststat = c("maximum", "quadratic", "scalar"),
              distribution = c("asymptotic", "approximate",
                               "exact", "none"),
              alternative = c("two.sided", "less", "greater"),
              xtrafo = trafo, ytrafo = trafo, scores = NULL,
              check = NULL, paired = FALSE, ...)
```

Arguments

formula	a formula of the form y1 + ... + yq ~ x | block where y1, ..., yq are measured on arbitrary scales (nominal, ordinal or continuous with or without censoring), x is a factor and block is an optional factor (which is generated automatically if omitted).
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations. (Not yet implemented!)
object	an object inheriting from classes "table" (with identical dimnames components) or "SymmetryProblem".
teststat	a character, the type of test statistic to be applied: either a maximum statistic ("maximum", default), a quadratic form ("quadratic") or a standardized scalar test statistic ("scalar").
distribution	a character, the conditional null distribution of the test statistic can be approximated by its asymptotic distribution ("asymptotic", default) or via Monte Carlo resampling ("approximate"). Alternatively, the functions asymptotic or approximate can be used. For univariate two-sample problems, "exact" or use of the function exact computes the exact distribution. Computation of the null distribution can be suppressed by specifying "none". It is also possible to specify a function with one argument (an object inheriting from "IndependenceTestStatistic") that returns an object of class "NullDistribution".
alternative	a character, the alternative hypothesis: either "two.sided" (default), "greater" or "less".
xtrafo	a function of transformations to be applied to the factor x supplied in formula; see 'Details'. Defaults to trafo().
ytrafo	a function of transformations to be applied to the variables y1, ..., yq supplied in formula; see 'Details'. Defaults to trafo().
scores	a named list of scores to be attached to ordered factors; see 'Details'. Defaults to NULL, implying equally spaced scores.

check	a function to be applied to objects of class "IndependenceTest" in order to check for specific properties of the data. Defaults to NULL.
paired	a logical, indicating that paired data have been transformed in such a way that the (unstandardized) linear statistic is the sum of the absolute values of the positive differences between the paired observations. Defaults to FALSE.
...	further arguments to be passed to or from other methods (currently ignored).

Details

symmetry_test() provides a general symmetry test for a set of variables measured on arbitrary scales. This function is based on the general framework for conditional inference procedures proposed by Strasser and Weber (1999). The salient parts of the Strasser-Weber framework are elucidated by Hothorn, Hornik, van de Wiel, and Zeileis (2006) and a thorough description of the software implementation is given by Hothorn, Hornik, van de Wiel, and Zeileis (2008).

The null hypothesis of symmetry is tested. The response variables and the measurement conditions are given by y1, ..., yq and x, respectively, and block is a factor where each level corresponds to exactly one subject with repeated measurements.

A vector of case weights, e.g., observation counts, can be supplied through the weights argument and the type of test statistic is specified by the teststat argument. Influence and regression functions, i.e., transformations of y1, ..., yq and x, are specified by the ytrafo and xtrafo arguments, respectively; see trafo() for the collection of transformation functions currently available. This allows for implementation of both novel and familiar test statistics, e.g., the McNemar test, the Cochran Q test, the Wilcoxon signed-rank test and the Friedman test. Furthermore, multivariate extensions such as the multivariate Friedman test (Gerig 1969; Puri and Sen 1971) can be implemented without much effort (see 'Examples').

If, say, y1 and/or x are ordered factors, the default scores, 1:nlevels(y1) and 1:nlevels(x), respectively, can be altered using the scores argument; this argument can also be used to coerce nominal factors to class "ordered". For example, when y1 is an ordered factor with four levels and x is a nominal factor with three levels, scores = list(y1 = c(1, 3:5), x = c(1:2, 4)) supplies the scores to be used. For ordered alternatives the scores must be monotonic, but non-monotonic scores are also allowed for testing against, e.g., umbrella alternatives. The length of the score vector must be equal to the number of factor levels.

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (distribution = "asymptotic"). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for univariate two-sample problems by setting distribution to "approximate" or "exact", respectively. See asymptotic(), approximate() and exact() for details.

Value

An object inheriting from class "IndependenceTest".

Note

Starting with version 1.1-0, maximum statistics and quadratic forms can no longer be specified using teststat = "maxtype" and teststat = "quadtype" respectively (as was used in versions prior to 0.4-5).

References

Gerig TM (1969). "A Multivariate Extension of Friedman's χ^2-Test." *Journal of the American Statistical Association*, **64**(328), 1595–1608. doi:10.1080/01621459.1969.10501079.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2006). "A Lego System for Conditional Inference." *The American Statistician*, **60**(3), 257–263. doi:10.1198/000313006X118430.

Hothorn T, Hornik K, van de Wiel MA, Zeileis A (2008). "Implementing a Class of Permutation Tests: The **coin** Package." *Journal of Statistical Software*, **28**(8), 1–23. doi:10.18637/jss.v028.i08.

Puri ML, Sen PK (1971). *Nonparametric Methods in Multivariate Analysis*. John Wiley & Sons, New York, U.S.A.

Strasser H, Weber C (1999). "On the Asymptotic Theory of Permutation Statistics." *Mathematical Methods of Statistics*, **8**(2), 220–250.

Examples

```
## One-sided exact Fisher-Pitman test for paired observations
y1 <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y2 <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
dta <- data.frame(
    y = c(y1, y2),
    x = gl(2, length(y1)),
    block = factor(rep(seq_along(y1), 2))
)

symmetry_test(y ~ x | block, data = dta,
              distribution = "exact", alternative = "greater")

## Alternatively: transform data and set 'paired = TRUE'
delta <- y1 - y2
y <- as.vector(rbind(abs(delta) * (delta >= 0), abs(delta) * (delta < 0)))
x <- factor(rep(0:1, length(delta)), labels = c("pos", "neg"))
block <- gl(length(delta), 2)

symmetry_test(y ~ x | block,
              distribution = "exact", alternative = "greater",
              paired = TRUE)


### Example data
### Gerig (1969, p. 1597)
gerig <- data.frame(
    y1 = c( 0.547, 1.811, 2.561,
            1.706, 2.509, 1.414,
           -0.288, 2.524, 3.310,
            1.417, 0.703, 0.961,
            0.878, 0.094, 1.682,
           -0.680, 2.077, 3.181,
            0.056, 0.542, 2.983,
            0.711, 0.269, 1.662,
           -1.335, 1.545, 2.920,
            1.635, 0.200, 2.065),
```

```
    y2 = c(-0.575, 1.840, 2.399,
            1.252, 1.574, 3.059,
           -0.310, 1.553, 0.560,
            0.932, 1.390, 3.083,
            0.819, 0.045, 3.348,
            0.497, 1.747, 1.355,
           -0.285, 0.760, 2.332,
            0.089, 1.076, 0.960,
           -0.349, 1.471, 4.121,
            0.845, 1.480, 3.391),
    x = factor(rep(1:3, 10)),
    b = factor(rep(1:10, each = 3))
)

### Asymptotic multivariate Friedman test
### Gerig (1969, p. 1599)
symmetry_test(y1 + y2 ~ x | b, data = gerig, teststat = "quadratic",
              ytrafo = function(data)
                  trafo(data, numeric_trafo = rank_trafo,
                        block = gerig$b)) # L_n = 17.238

### Asymptotic multivariate Page test
(st <- symmetry_test(y1 + y2 ~ x | b, data = gerig,
                     ytrafo = function(data)
                         trafo(data, numeric_trafo = rank_trafo,
                               block = gerig$b),
                     scores = list(x = 1:3)))
pvalue(st, method = "step-down")
```

SymmetryTests *Symmetry Tests*

Description

Testing the symmetry of a numeric repeated measurements variable in a complete block design.

Usage

```
## S3 method for class 'formula'
sign_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'SymmetryProblem'
sign_test(object, ...)

## S3 method for class 'formula'
wilcoxsign_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'SymmetryProblem'
wilcoxsign_test(object, zero.method = c("Pratt", "Wilcoxon"), ...)

## S3 method for class 'formula'
```

```
friedman_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'SymmetryProblem'
friedman_test(object, ...)

## S3 method for class 'formula'
quade_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'SymmetryProblem'
quade_test(object, ...)
```

Arguments

formula	a formula of the form y ~ x | block where y is a numeric variable, x is a factor with two (sign_test and wilcoxsign_test) or more levels and block is an optional factor (which is generated automatically if omitted).
data	an optional data frame containing the variables in the model formula.
subset	an optional vector specifying a subset of observations to be used. Defaults to NULL.
weights	an optional formula of the form ~ w defining integer valued case weights for each observation. Defaults to NULL, implying equal weight for all observations. (Not yet implemented!)
object	an object inheriting from class "SymmetryProblem".
zero.method	a character, the method used to handle zeros: either "Pratt" (default) or "Wilcoxon"; see 'Details'.
...	further arguments to be passed to symmetry_test().

Details

sign_test(), wilcoxsign_test(), friedman_test() and quade_test() provide the sign test, the Wilcoxon signed-rank test, the Friedman test, the Page test and the Quade test Quade (1979). A general description of these methods is given by Hollander and Wolfe (1999).

The null hypothesis of symmetry is tested. The response variable and the measurement conditions are given by y and x, respectively, and block is a factor where each level corresponds to exactly one subject with repeated measurements. For sign_test and wilcoxsign_test, formulae of the form y ~ x | block and y ~ x are allowed. The latter form is interpreted as y is the first and x the second measurement on the same subject.

If x is an ordered factor, the default scores, 1:nlevels(x), can be altered using the scores argument (see symmetry_test()); this argument can also be used to coerce nominal factors to class "ordered". In this case, a linear-by-linear association test is computed and the direction of the alternative hypothesis can be specified using the alternative argument. For the Friedman test, this extension was given by Page (1963) and is known as the Page test.

For wilcoxsign_test(), the default method of handling zeros (zero.method = "Pratt"), due to Pratt (1959), first rank-transforms the absolute differences (including zeros) and then discards the ranks corresponding to the zero-differences. The proposal by Wilcoxon (1949, p. 6) first discards the zero-differences and then rank-transforms the remaining absolute differences (zero.method = "Wilcoxon").

The conditional null distribution of the test statistic is used to obtain p-values and an asymptotic approximation of the exact distribution is used by default (`distribution = "asymptotic"`). Alternatively, the distribution can be approximated via Monte Carlo resampling or computed exactly for univariate two-sample problems by setting `distribution` to `"approximate"` or `"exact"`, respectively. See `asymptotic()`, `approximate()` and `exact()` for details.

Value

An object inheriting from class `"IndependenceTest"`.

Note

Starting with version 1.0-16, the `zero.method` argument replaced the (now removed) `ties.method` argument. The current default is `zero.method = "Pratt"` whereas earlier versions had `ties.method = "HollanderWolfe"`, which is equivalent to `zero.method = "Wilcoxon"`.

References

Hollander M, Wolfe DA (1999). *Nonparametric Statistical Inference*, 2nd edition. John Wiley & Sons, New York, U.S.A.

Page EB (1963). "Ordered Hypotheses for Multiple Treatments: A Significance Test for Linear Ranks." *Journal of the American Statistical Association*, **58**(301), 216–230. doi:10.1080/01621459.1963.10500843.

Pratt JW (1959). "Remarks on Zeros and Ties in the Wilcoxon Signed Rank Procedures." *Journal of the American Statistical Association*, **54**(287), 655–667. doi:10.1080/01621459.1959.10501526.

Quade D (1979). "Using Weighted Rankings in the Analysis of Complete Blocks with Additive Block Effects." *Journal of the American Statistical Association*, **74**(367), 680–683. doi:10.1080/01621459.1979.10481670.

Wilcoxon F (1949). "Some Rapid Approximate Statistical Procedures." American Cyanamid Company, New York, U.S.A.

Examples

```
## Example data from ?wilcox.test
y1 <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y2 <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)

## One-sided exact sign test
(st <- sign_test(y1 ~ y2, distribution = "exact",
                 alternative = "greater"))
midpvalue(st) # mid-p-value

## One-sided exact Wilcoxon signed-rank test
(wt <- wilcoxsign_test(y1 ~ y2, distribution = "exact",
                       alternative = "greater"))
statistic(wt, type = "linear")
midpvalue(wt) # mid-p-value

## Comparison with R's wilcox.test() function
```

```
wilcox.test(y1, y2, paired = TRUE, alternative = "greater")


## Data with explicit group and block information
dta <- data.frame(y = c(y1, y2), x = gl(2, length(y1)),
                  block = factor(rep(seq_along(y1), 2)))

## For two samples, the sign test is equivalent to the Friedman test...
sign_test(y ~ x | block, data = dta, distribution = "exact")
friedman_test(y ~ x | block, data = dta, distribution = "exact")

## ...and the signed-rank test is equivalent to the Quade test
wilcoxsign_test(y ~ x | block, data = dta, distribution = "exact")
quade_test(y ~ x | block, data = dta, distribution = "exact")


## Comparison of three methods ("round out", "narrow angle", and "wide angle")
## for rounding first base.
## Hollander and Wolfe (1999, p. 274, Tab. 7.1)
rounding <- data.frame(
    times = c(5.40, 5.50, 5.55,
              5.85, 5.70, 5.75,
              5.20, 5.60, 5.50,
              5.55, 5.50, 5.40,
              5.90, 5.85, 5.70,
              5.45, 5.55, 5.60,
              5.40, 5.40, 5.35,
              5.45, 5.50, 5.35,
              5.25, 5.15, 5.00,
              5.85, 5.80, 5.70,
              5.25, 5.20, 5.10,
              5.65, 5.55, 5.45,
              5.60, 5.35, 5.45,
              5.05, 5.00, 4.95,
              5.50, 5.50, 5.40,
              5.45, 5.55, 5.50,
              5.55, 5.55, 5.35,
              5.45, 5.50, 5.55,
              5.50, 5.45, 5.25,
              5.65, 5.60, 5.40,
              5.70, 5.65, 5.55,
              6.30, 6.30, 6.25),
    methods = factor(rep(1:3, 22),
                     labels = c("Round Out", "Narrow Angle", "Wide Angle")),
    block = gl(22, 3)
)

## Asymptotic Friedman test
friedman_test(times ~ methods | block, data = rounding)

## Parallel coordinates plot
with(rounding, {
    matplot(t(matrix(times, ncol = 3, byrow = TRUE)),
```

```
            type = "l", lty = 1, col = 1, ylab = "Time", xlim = c(0.5, 3.5),
            axes = FALSE)
    axis(1, at = 1:3, labels = levels(methods))
    axis(2)
})

## Where do the differences come from?
## Wilcoxon-Nemenyi-McDonald-Thompson test (Hollander and Wolfe, 1999, p. 295)
## Note: all pairwise comparisons
(st <- symmetry_test(times ~ methods | block, data = rounding,
                     ytrafo = function(data)
                         trafo(data, numeric_trafo = rank_trafo,
                               block = rounding$block),
                     xtrafo = mcp_trafo(methods = "Tukey")))

## Simultaneous test of all pairwise comparisons
## Wide Angle vs. Round Out differ (Hollander and Wolfe, 1999, p. 296)
pvalue(st, method = "single-step") # subset pivotality is violated


## Strength Index of Cotton
## Hollander and Wolfe (1999, p. 286, Tab. 7.5)
cotton <- data.frame(
    strength = c(7.46, 7.17, 7.76, 8.14, 7.63,
                 7.68, 7.57, 7.73, 8.15, 8.00,
                 7.21, 7.80, 7.74, 7.87, 7.93),
    potash = ordered(rep(c(144, 108, 72, 54, 36), 3),
                     levels = c(144, 108, 72, 54, 36)),
    block = gl(3, 5)
)

## One-sided asymptotic Page test
friedman_test(strength ~ potash | block, data = cotton, alternative = "greater")

## One-sided approximative (Monte Carlo) Page test
friedman_test(strength ~ potash | block, data = cotton, alternative = "greater",
              distribution = approximate(nresample = 10000))


## Data from Quade (1979, p. 683)
dta <- data.frame(
    y = c(52, 45, 38,
          63, 79, 50,
          45, 57, 39,
          53, 51, 43,
          47, 50, 56,
          62, 72, 49,
          49, 52, 40),
     x = factor(rep(LETTERS[1:3], 7)),
     b = factor(rep(1:7, each = 3))
)

## Approximative (Monte Carlo) Friedman test
```

```
## Quade (1979, p. 683)
friedman_test(y ~ x | b, data = dta,
              distribution = approximate(nresample = 10000)) # chi^2 = 6.000

## Approximative (Monte Carlo) Quade test
## Quade (1979, p. 683)
(qt <- quade_test(y ~ x | b, data = dta,
                  distribution = approximate(nresample = 10000))) # W = 8.157

## Comparison with R's quade.test() function
quade.test(y ~ x | b, data = dta)

## quade.test() uses an F-statistic
b <- nlevels(qt@statistic@block)
A <- sum(qt@statistic@ytrans^2)
B <- sum(statistic(qt, type = "linear")^2) / b
(b - 1) * B / (A - B) # F = 8.3765
```

Transformations *Functions for Data Transformation*

Description

Transformations for factors and numeric variables.

Usage

```
id_trafo(x)
rank_trafo(x, ties.method = c("mid-ranks", "random"))
normal_trafo(x, ties.method = c("mid-ranks", "average-scores"))
median_trafo(x, mid.score = c("0", "0.5", "1"))
savage_trafo(x, ties.method = c("mid-ranks", "average-scores"))
consal_trafo(x, ties.method = c("mid-ranks", "average-scores"), a = 5)
koziol_trafo(x, ties.method = c("mid-ranks", "average-scores"), j = 1)
klotz_trafo(x, ties.method = c("mid-ranks", "average-scores"))
mood_trafo(x, ties.method = c("mid-ranks", "average-scores"))
ansari_trafo(x, ties.method = c("mid-ranks", "average-scores"))
fligner_trafo(x, ties.method = c("mid-ranks", "average-scores"))
logrank_trafo(x, ties.method = c("mid-ranks", "Hothorn-Lausen",
                                 "average-scores"),
              weight = logrank_weight, ...)
logrank_weight(time, n.risk, n.event,
               type = c("logrank", "Gehan-Breslow", "Tarone-Ware",
                        "Peto-Peto", "Prentice", "Prentice-Marek",
                        "Andersen-Borgan-Gill-Keiding", "Fleming-Harrington",
                        "Gaugler-Kim-Liao", "Self"),
               rho = NULL, gamma = NULL)
f_trafo(x)
```

```
of_trafo(x, scores = NULL)
zheng_trafo(x, increment = 0.1)
maxstat_trafo(x, minprob = 0.1, maxprob = 1 - minprob)
fmaxstat_trafo(x, minprob = 0.1, maxprob = 1 - minprob)
ofmaxstat_trafo(x, minprob = 0.1, maxprob = 1 - minprob)
trafo(data, numeric_trafo = id_trafo, factor_trafo = f_trafo,
      ordered_trafo = of_trafo, surv_trafo = logrank_trafo,
      var_trafo = NULL, block = NULL)
mcp_trafo(...)
```

Arguments

`x`	an object of class `"numeric"`, `"factor"`, `"ordered"` or `"Surv"`.
`ties.method`	a character, the method used to handle ties. The score generating function either uses the mid-ranks (`"mid-ranks"`, default) or, in the case of `rank_trafo()`, randomly broken ties (`"random"`). Alternatively, the average of the scores resulting from applying the score generating function to randomly broken ties are used (`"average-scores"`). See `logrank_test()` for a detailed description of the methods used in `logrank_trafo()`.
`mid.score`	a character, the score assigned to observations exactly equal to the median: either 0 (`"0"`, default), 0.5 (`"0.5"`) or 1 (`"1"`); see `median_test()`.
`a`	a numeric vector, the values taken as the constant a in the Conover-Salsburg scores. Defaults to 5.
`j`	a numeric, the value taken as the constant j in the Koziol-Nemec scores. Defaults to 1.
`weight`	a function where the first three arguments must correspond to `time`, `n.risk`, and `n.event` given below. Defaults to `logrank_weight`.
`time`	a numeric vector, the ordered distinct time points.
`n.risk`	a numeric vector, the number of subjects at risk at each time point specified in `time`.
`n.event`	a numeric vector, the number of events at each time point specified in `time`.
`type`	a character, one of `"logrank"` (default), `"Gehan-Breslow"`, `"Tarone-Ware"`, `"Peto-Peto"`, `"Prentice"`, `"Prentice-Marek"`, `"Andersen-Borgan-Gill-Keiding"`, `"Fleming-Harrington"`, `"Gaugler-Kim-Liao"` or `"Self"`; see `logrank_test()`.
`rho`	a numeric vector, the ρ constant when `type` is `"Tarone-Ware"`, `"Fleming-Harrington"`, `"Gaugler-Kim-Liao"` or `"Self"`; see `logrank_test()`. Defaults to `NULL`, implying `0.5` for `type = "Tarone-Ware"` and `0` otherwise.
`gamma`	a numeric vector, the γ constant when `type` is `"Fleming-Harrington"`, `"Gaugler-Kim-Liao"` or `"Self"`; see `logrank_test()`. Defaults to `NULL`, implying `0`.
`scores`	a numeric vector or list, the scores corresponding to each level of an ordered factor. Defaults to `NULL`, implying `1:nlevels(x)`.
`increment`	a numeric, the score increment between the order-restricted sets of scores. A fraction greater than 0, but smaller than or equal to 1. Defaults to `0.1`.
`minprob`	a numeric, a fraction between 0 and 0.5; see `maxstat_test()`. Defaults to `0.1`.

maxprob	a numeric, a fraction between 0.5 and 1; see maxstat_test(). Defaults to 1 - minprob.
data	an object of class "data.frame".
numeric_trafo	a function to be applied to elements of class "numeric" in data, returning a matrix with nrow(data) rows and an arbitrary number of columns. Defaults to id_trafo.
factor_trafo	a function to be applied to elements of class "factor" in data, returning a matrix with nrow(data) rows and an arbitrary number of columns. Defaults to f_trafo.
ordered_trafo	a function to be applied to elements of class "ordered" in data, returning a matrix with nrow(data) rows and an arbitrary number of columns. Defaults to of_trafo.
surv_trafo	a function to be applied to elements of class "Surv" in data, returning a matrix with nrow(data) rows and an arbitrary number of columns. Defaults to logrank_trafo.
var_trafo	an optional named list of functions to be applied to the corresponding variables in data. Defaults to NULL.
block	an optional factor whose levels are interpreted as blocks. trafo is applied to each level of block separately. Defaults to NULL.
...	logrank_trafo(): further arguments to be passed to weight. mcp_trafo(): factor name and contrast matrix (as matrix or character) in a 'tag = value' format for multiple comparisons based on a single unordered factor; see mcp() in package **multcomp**.

Details

The utility functions documented here are used to define specialized test procedures.

id_trafo() is the identity transformation.

rank_trafo(), normal_trafo(), median_trafo(), savage_trafo(), consal_trafo() and koziol_trafo() compute rank (Wilcoxon) scores, normal (van der Waerden) scores, median (Mood-Brown) scores, Savage scores, Conover-Salsburg scores (see neuropathy) and Koziol-Nemec scores, respectively, for location problems.

klotz_trafo(), mood_trafo(), ansari_trafo() and fligner_trafo() compute Klotz scores, Mood scores, Ansari-Bradley scores and Fligner-Killeen scores, respectively, for scale problems.

logrank_trafo() computes weighted logrank scores for right-censored data, allowing for a user-defined weight function through the weight argument (see GTSG).

f_trafo() computes dummy matrices for factors and of_trafo() assigns scores to ordered factors. For ordered factors with two levels, the scores are normalized to the $[0, 1]$ range. zheng_trafo() computes a finite collection of order-restricted scores for ordered factors (see jobsatisfaction, malformations and vision).

maxstat_trafo(), fmaxstat_trafo() and ofmaxstat_trafo() compute scores for cutpoint problems (see maxstat_test()).

trafo() applies its arguments to the elements of data according to the classes of the elements. A trafo() function with modified default arguments is usually supplied to independence_test()

via the xtrafo or ytrafo arguments. Fine tuning, i.e., different transformations for different variables, is possible by supplying a named list of functions to the var_trafo argument.

mcp_trafo() computes contrast matrices for factors.

Value

A numeric vector or matrix with nrow(x) rows and an arbitrary number of columns. For trafo(), a named matrix with nrow(data) rows and an arbitrary number of columns.

Note

Starting with version 1.1-0, all transformation functions are now passing through missing values (i.e., NAs). Furthermore, median_trafo() and logrank_trafo() are now *increasing* functions (for consistency with most other transformations in this package).

Examples

```
## Dummy matrix, two-sample problem (only one column)
f_trafo(gl(2, 3))

## Dummy matrix, K-sample problem (K columns)
x <- gl(3, 2)
f_trafo(x)

## Score matrix
ox <- as.ordered(x)
of_trafo(ox)
of_trafo(ox, scores = c(1, 3:4))
of_trafo(ox, scores = list(s1 = 1:3, s2 = c(1, 3:4)))
zheng_trafo(ox, increment = 1/3)

## Normal scores
y <- runif(6)
normal_trafo(y)

## All together now
trafo(data.frame(x = x, ox = ox, y = y), numeric_trafo = normal_trafo)

## The same, but allows for fine-tuning
trafo(data.frame(x = x, ox = ox, y = y), var_trafo = list(y = normal_trafo))

## Transformations for maximally selected statistics
maxstat_trafo(y)
fmaxstat_trafo(x)
ofmaxstat_trafo(ox)

## Apply transformation blockwise (as in the Friedman test)
trafo(data.frame(y = 1:20), numeric_trafo = rank_trafo, block = gl(4, 5))

## Multiple comparisons
dta <- data.frame(x)
mcp_trafo(x = "Tukey")(dta)
```

```
## The same, but useful when specific contrasts are desired
K <- rbind("2 - 1" = c(-1,  1, 0),
           "3 - 1" = c(-1,  0, 1),
           "3 - 2" = c( 0, -1, 1))
mcp_trafo(x = K)(dta)
```

`treepipit` *Tree Pipits in Franconian Oak Forests*

Description

Data on the population density of tree pipits, *Anthus trivialis*, in Franconian oak forests including variables describing the forest ecosystem.

Usage

```
treepipit
```

Format

A data frame with 86 observations on 10 variables.

`counts` the number of tree pipits observed.

`age` age of the overstorey oaks taken from forest data.

`coverstorey` cover of canopy overstorey (%). The crown cover is described relative to a fully stocked stand. Very dense overstorey with multiple crown cover could reach values greater than 100%.

`coverregen` cover of regeneration and shrubs (%).

`meanregen` mean height of regeneration and shrubs.

`coniferous` coniferous trees (% per hectare).

`deadtree` number of dead trees (per hectare).

`cbpiles` number of crowns and branch piles (per hectare). All laying crowns and branch piles were counted. These were induced by logging and the creation of wind breaks.

`ivytree` number of ivied trees (per hectare).

`fdist` distance to the forest edge. The closest distance to the forest edge was measured from the centre of each grid.

Details

This study (Müller and Hothorn 2004) is based on fieldwork conducted in three lowland oak forests in the Franconian region of northern Bavaria close to Uffenheim, Germany. Diurnal breeding birds were sampled five times, from March to June 2002, using a quantitative grid mapping. Each grid was a one-hectare square. In total, 86 sample sites were established in 9 stands. All individuals were counted in time intervals of 7 min/grid during slow walks along the middle of the grid with a stop in the centre. Environmental factors were measured for each grid.

Source

Müller J, Hothorn T (2004). "Maximally Selected Two-Sample Statistics as a New Tool for the Identification and Assessment of Habitat Factors with an Application to Breeding Bird Communities in Oak Forests." *European Journal of Forest Research*, **123**, 218–228. doi:10.1007/s10342004-00355.

Examples

```
## Asymptotic maximally selected statistics
maxstat_test(counts ~ age + coverstorey + coverregen + meanregen +
                      coniferous + deadtree + cbpiles + ivytree,
             data = treepipit)
```

VarCovar-class *Class* "VarCovar" *and its subclasses*

Description

Objects of class "VarCovar" and its subclasses "CovarianceMatrix" and "Variance" represent the covariance and variance, respectively, of the linear statistic.

Objects from the Class

Class "VarCovar" is a *virtual* class defined as the class union of "CovarianceMatrix" and "Variance", so objects cannot be created from it directly.

Objects can be created by calls of the form

```
new("CovarianceMatrix", covariance, \dots)
```

and

```
new("Variance", variance, \dots)
```

where covariance is a covariance matrix and variance is numeric vector containing the diagonal elements of the covariance matrix.

Slots

For objects of class "CovarianceMatrix":

covariance: Object of class "matrix". The covariance matrix.

For objects of class "Variance":

variance: Object of class "numeric". The diagonal elements of the covariance matrix.

Extends

For objects of classes "CovarianceMatrix" or "Variance":
Class "VarCovar", directly.

Known Subclasses

For objects of class "VarCovar":
Class "CovarianceMatrix", directly.
Class "Variance", directly.

Methods

covariance signature(object = "CovarianceMatrix"): See the documentation for covariance for details.

initialize signature(.Object = "CovarianceMatrix"): See the documentation for initialize (in package **methods**) for details.

initialize signature(.Object = "Variance"): See the documentation for initialize (in package **methods**) for details.

variance signature(object = "CovarianceMatrix"): See the documentation for variance for details.

variance signature(object = "Variance"): See the documentation for variance for details.

Note

Starting with version 1.4-0, this class is deprecated. **It will be made defunct and removed in a future release.**

---

vision *Unaided Distance Vision*

Description

Assessment of unaided distance vision of women in Britain.

Usage

```
vision
```

Format

A contingency table with 7477 observations on 2 variables.

Right.Eye a factor with levels "Highest Grade", "Second Grade", "Third Grade" and "Lowest Grade".

Left.Eye a factor with levels "Highest Grade", "Second Grade", "Third Grade" and "Lowest Grade".

Details

Paired ordered categorical data from case-records of eye-testing of 7477 women aged 30–39 years employed by Royal Ordnance Factories in Britain during 1943–46, as given by Stuart (1955).

This data set was used by Stuart (1955) to illustrate a test of marginal homogeneity. Winell and Lindbäck (2018) also used the data, demonstrating a score-independent test for ordered categorical data.

References

Stuart A (1955). "A Test for Homogeneity of the Marginal Distributions in a Two-way Classification." *Biometrika*, **42**(3–4), 412–416. doi:10.1093/biomet/42.34.412.

Winell H, Lindbäck J (2018). "A General Score-independent Test For Order-restricted Inference." *Statistics in Medicine*, **37**(21), 3078–3090. doi:10.1002/sim.7690.

Examples

```
## Asymptotic Stuart test (Q = 11.96)
diag(vision) <- 0 # speed-up
mh_test(vision)

## Asymptotic score-independent test
## Winell and Lindbaeck (2018)
(st <- symmetry_test(vision,
                     ytrafo = function(data)
                         trafo(data, factor_trafo = function(y)
                             zheng_trafo(as.ordered(y)))))
ss <- statistic(st, type = "standardized")
idx <- which(abs(ss) == max(abs(ss)), arr.ind = TRUE)
ss[idx[1], idx[2], drop = FALSE]
```

Index