

Package ‘coin’

October 30, 2009

Title Conditional Inference Procedures in a Permutation Test Framework

Date 2009-10-30

Version 1.0-8

Author Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel and Achim Zeileis

Maintainer Torsten Hothorn <Torsten.Hothorn@R-project.org>

Description Conditional inference procedures for the general independence problem including two-sample, K-sample (non-parametric ANOVA), correlation, censored, ordered and multivariate problems.

Depends R (>= 2.2.0), methods, survival, mvtnorm (>= 0.8-0), modeltools (>= 0.2-9)

Suggests multcomp, xtable, e1071, vcd

Enhances Biobase

LazyLoad yes

LazyData yes

License GPL-2

Repository CRAN

Date/Publication 2009-10-30 17:49:07

R topics documented:

alpha	2
alzheimer	3
asat	4
coin	5
ContingencyTests	6
Distribution	9
ExactNullDistribution-methods	10
expectation-methods	10

ExpectCovar-class	11
ExpectCovarInfluence-class	12
glioma	12
hohnloser	14
IndependenceLinearStatistic-class	15
IndependenceProblem-class	16
IndependenceTest	16
IndependenceTest-class	19
IndependenceTestProblem-class	20
IndependenceTestStatistic-class	20
initialize-methods	21
jobsatisfaction	22
LocationTests	22
MarginalHomogeneityTest	26
MaxstatTest	28
MaxTypeIndependenceTestStatistic-class	30
mercuryfish	31
neuropathy	33
NullDistribution-class	34
ocarcinoma	35
PermutationDistribution	36
photocar	37
PValue-class	38
pvalue-methods	39
QuadTypeIndependenceTestStatistic-class	40
rotarod	41
ScalarIndependenceTestStatistic-class	42
ScaleTests	43
show-methods	46
SpearmanTest	47
sphase	48
statistic-methods	49
SurvTest	50
SymmetryProblem-class	52
SymmetryTest	53
SymmetryTests	54
Transformations	57
treepipit	59

Description

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

Usage

```
data("alpha")
```

Format

A data frame with 97 observations on the following 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 et al. (2005) found longer alleles of NACP-REP1 in alcohol-dependent patients compared with healthy controls and report that the allele lengths show some association with levels of expressed alpha synuclein mRNA.

Source

Dominikus Bönsch, Thomas Lederer, Udo Reulbach, Torsten Hothorn, Johannes Kornhuber & Stefan Bleich (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.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2006). A Lego system for conditional inference, *The American Statistician*, **60**(3), 257–263.

Examples

```
boxplot(elevel ~ alength, data = alpha)
kruskal_test(elevel ~ alength, data = alpha)
```

 alzheimer

Smoking and Alzheimer's Disease

Description

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

Usage

```
data("alzheimer")
```

Format

A data frame with 538 observations on the following 3 variables.

smoking a factor at levels None, <10, 10–20 and >20 (cigarettes per day).

disease a factor at levels Alzheimer, Other dementias and Other diagnoses.

gender a factor at levels Female and Male.

Details

198 cases of Alzheimer's disease are compared to a control group with respect to smoking history. The data are published in Table 4 (Salib & Hillier, 1997).

Source

Emad Salib & Valerie Hillier (1997). A case-control study of smoking and Alzheimer's disease, *International Journal of Geriatric Psychiatry* **12**, 295–300.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2006). A Lego system for conditional inference, *The American Statistician*, **60**(3), 257–263.

Examples

```
### spineplots
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")

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

asat

Toxicological Study on Female Wistar Rats

Description

ASAT-values for a new compound and a control group of 34 female Wistar rats.

Usage

```
data("asat")
```

Format

A data frame with 34 observations on the following 2 variables.

asat the ASAT-values (a liver enzyme)

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 are originally given in Hothorn (1992) and reproduced in Hauschke et al. (1999).

Source

Ludwig A. Hothorn (1992), Biometrische Analyse toxikologischer Untersuchungen. In: J. Adams (ed.): *Statistisches Know how in der medizinischen Forschung*. Ullstein-Mosby, Berlin, 475–590.

References

Dieter Hauschke, Meinhard Kieser & Ludwig A. Hothorn (1999). Proof of safety in toxicology based on the ratio of two means for normally distributed data, *Biometrical Journal* **41**(3), 295–304.

Rafael Pflüger & Torsten Hothorn (2002). Assessing equivalence tests with respect to their expected *p*-Value, *Biometrical Journal* **44**(8), 1002–1027.

Examples

```
### proof-of-safety based on ratio of medians
pos <- wilcox_test(I(log(asat)) ~ group, data = asat, alternative = "less",
                  conf.int = TRUE, distribution = "exact")

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

Description

The **coin** package implements a general framework for conditional inference procedures, commonly known as *permutation tests*, theoretically derived by Strasser & Weber (1999). The conditional expectation and covariance for a broad class of multivariate linear statistics as well as the corresponding multivariate limiting distribution was derived by Strasser & Weber (1999). These results are utilized to construct tests for independence between two sets of variables.

Beside a general implementation of the abstract framework the package offers a rather huge set of convenience functions implementing well known classical as well as less prominent classical and non-classical test procedures in a conditional inference framework. Examples are 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 for independence of two continuous variables as well as tests for marginal homogeneity and symmetry. Conditional counterparts of most of the classical procedures given in famous text books like Hollander & Wolfe (1999) or Agresti (2002) can be implemented as part of the general framework without much effort. Approximations of the exact null distribution via the limiting distribution and conditional Monte-Carlo procedures are available for every test while the exact null distribution is currently available for two-sample problems only.

Author(s)

This package is written by
Torsten Hothorn <Torsten.Hothorn@R-project.org>,
Kurt Hornik <Kurt.Hornik@R-project.org>,
Mark van de Wiel <m.a.v.d.wiel@TUE.nl> and
Achim Zeileis <Achim.Zeileis@R-project.org>.

References

- Helmut Strasser & Christian Weber (1999). On the asymptotic theory of permutation statistics, *Mathematical Methods of Statistics* **8**, 220–250.
- Myles Hollander & Douglas A. Wolfe (1999). *Nonparametric Statistical Methods, 2nd Edition*. New York: John Wiley & Sons.
- Alan Agresti (2002). *Categorical Data Analysis* Hoboken, New Jersey: John Wiley & Sons.
- Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2006). A Lego system for conditional inference, *The American Statistician*, **60**(3), 257–263.
- Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2008). Implementing a class of permutation tests: The coin package, *Journal of Statistical Software*, **28**(8), 1–23. <http://www.jstatsoft.org/v28/i08/>

Description

Testing the independence of two possibly ordered factors, eventually stratified by a third factor.

Usage

```
## S3 method for class 'formula':
cmh_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table':
cmh_test(object, distribution = c("asymptotic", "approximate"), ...)
## S3 method for class 'IndependenceProblem':
cmh_test(object, distribution = c("asymptotic", "approximate"), ...)

## S3 method for class 'formula':
chisq_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table':
chisq_test(object, distribution = c("asymptotic", "approximate"), ...)
## S3 method for class 'IndependenceProblem':
chisq_test(object, distribution = c("asymptotic", "approximate"), ...)

## S3 method for class 'formula':
lbl_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'table':
lbl_test(object, distribution = c("asymptotic", "approximate"), ...)
## S3 method for class 'IndependenceProblem':
lbl_test(object, distribution = c("asymptotic", "approximate"), ...)
```

Arguments

<code>formula</code>	a formula of the form $y \sim x \mid \text{block}$ where y and x are factors (possibly ordered) and <code>block</code> is an optional factor for stratification.
<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>weights</code>	an optional formula of the form $\sim w$ defining integer valued weights for the observations.
<code>object</code>	an object inheriting from class "IndependenceProblem" or an object of class <code>table</code> .
<code>distribution</code>	a character, the null distribution of the test statistic can be approximated by its asymptotic distribution ("asymptotic") or via Monte-Carlo resampling ("approximate"). Alternatively, the functions approximate or asymptotic


```

distribution = approximate(B = 10000)

### Smoking and HDL cholesterol status
### (from Jeong, Jhun and Kim, 2005, CSDA 48, 623-631, Table 2)
smokingHDL <- as.table(
  matrix(c(15, 8, 11, 5,
           3, 4, 6, 1,
           6, 7, 15, 11,
           1, 2, 3, 5), ncol = 4,
         dimnames = list(smoking = c("none", "< 5", "< 10", ">=10"),
                        HDL = c("normal", "low", "borderline", "abnormal")))
)
### use interval mid-points as scores for smoking
lbl_test(smokingHDL, scores = list(smoking = c(0, 2.5, 7.5, 15)))

### Cochran-Armitage trend test for proportions
### Lung tumors in female mice exposed to 1,2-dichloroethane
### Encyclopedia of Biostatistics (Armitage & Colton, 1998),
### Chapter Trend Test for Counts and Proportions, page 4578, Table 2
lungtumor <- data.frame(dose = rep(c(0, 1, 2), c(40, 50, 48)),
                       tumor = c(rep(c(0, 1), c(38, 2)),
                                 rep(c(0, 1), c(43, 7)),
                                 rep(c(0, 1), c(33, 15))))
table(lungtumor$dose, lungtumor$tumor)

### Cochran-Armitage test (permutation equivalent to correlation
### between dose and tumor), cf. Table 2 for results
independence_test(tumor ~ dose, data = lungtumor, teststat = "quad")

### linear-by-linear association test with scores 0, 1, 2
### is identical with Cochran-Armitage test
lungtumor$dose <- ordered(lungtumor$dose)
independence_test(tumor ~ dose, data = lungtumor, teststat = "quad",
                 scores = list(dose = c(0, 1, 2)))

```

Distribution

Distribution under the Null Hypothesis

Description

Specification of the exact, or approximation of the exact, conditional distribution of test statistics under the null hypothesis.

Usage

```

exact(algorithm = c("shift", "split-up"), fact = NULL)
approximate(B = 1000)
asymptotic(maxpts = 25000, abseps = 0.001, releps = 0)

```

Arguments

algorithm	a character, specifying the algorithm to be used for the computation of the exact conditional distribution.
fact	a positive integer to multiply the response values with.
B	a positive integer, the number of Monte-Carlo replications to approximate the exact conditional distribution.
maxpts	a positive integer, the maximum number of function values, see pmvnorm .
abseps	double, the absolute error tolerance, see pmvnorm .
releps	double, relative error tolerance, see pmvnorm .

Details

The `distribution` argument to [independence_test](#) can be specified with additional arguments using those functions.

Exact algorithms are currently only implemented for two-sample problems.

Value

An object of class `exact`, `approximate` or `asymptotic`, respectively.

ExactNullDistribution-methods

Methods for computing exact conditional reference distributions

Description

~~ Methods for function `ExactNullDistribution` in Package 'coin' ~~

Methods

object = "ScalarIndependenceTestStatistic" ~~describe this method here

expectation-methods

Extract the Expectation and Variance / Covariance of Linear Statistics

Description

Extracts the conditional expectation and covariance for linear statistics from objects inheriting from "IndependenceTest".

Usage

```
expectation(object, ...)  
covariance(object, ...)  
variance(object, ...)
```

Arguments

`object` an object inheriting from class `IndependenceTest-class`.
`...` further arguments (currently ignored).

Methods

expectation extracts the expectation of the linear statistic of `object`.

covariance extracts the covariance of the linear statistic of `object`.

variance extracts the variance of the linear statistic of `object`.

Examples

```
df <- data.frame(y = gl(3, 2), x = gl(3, 2)[sample(1:6)])  
  
### Cochran-Mantel-Haenzel Test  
ct <- cmh_test(y ~ x, data = df)  
  
### the linear statistic, i.e, the contingency table  
l <- statistic(ct, type = "linear")  
l  
  
### expectation  
E1 <- expectation(ct)  
E1  
  
### covariance  
V1 <- covariance(ct)  
V1  
  
### the standardized contingency table (hard way)  
(l - E1) / sqrt(variance(ct))  
  
### easy way  
statistic(ct, type = "standardized")
```

ExpectCovar-class *Class "ExpectCovar"*

Description

Conditional expectation and covariance of linear statistics

Objects from the Class

Objects can be created by calls of the form `new ("ExpectCovar", ...)`. Normally, this class is used internally only.

Slots

expectation: Object of class "numeric"

covariance: Object of class "matrix"

dimension: Object of class "integer"

ExpectCovarInfluence-class
 Class "ExpectCovarInfluence"

Description

Conditional expectation and covariance of influence functions

Objects from the Class

Objects can be created by calls of the form `new ("ExpectCovarInfluence", ...)`. Normally, this class is used internally only.

Slots

sumweights: Object of class "numeric".

expectation: Object of class "numeric".

covariance: Object of class "matrix".

dimension: Object of class "integer".

Extends

Class "ExpectCovar", directly.

glioma

Malignant Glioma Pilot Study

Description

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

Usage

```
data("glioma")
```

Format

A data frame with 37 observations on the following 7 variables.

no. patient number.

age patients ages in years.

sex a factor with levels F(emale) and M(ale).

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

time survival times in month.

event censoring indicator: FALSE censored and TRUE dead.

group a factor with levels Control and RIT.

Details

The primary endpoint of this small pilot study is survival. Survival times are tied, the usual asymptotic log-rank test may be inadequate in this setup. Therefore, a permutation test (via Monte-Carlo sampling) was conducted in the original paper. The data are taken from Tables 1 and 2 of Grana et al. (2002).

Source

C. Grana, M. Chinol, C. Robertson, C. Mazzetta, M. Bartolomei, C. De Cicco, M. Fiorenza, M. Gatti, P. Caliceti & G. Paganelli (2002), Pretargeted adjuvant radioimmunotherapy with Yttrium-90-biotin in malignant glioma patients: A pilot study. *British Journal of Cancer* **86**(2), 207–212.

Examples

```
layout(matrix(1:2, ncol = 2))

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

### Plot Kaplan-Meier curves
plot(survfit(Surv(time, event) ~ group, data = g3),
```

```

    main = "Grade III Glioma", lty = c(2,1),
    legend.text = c("Control", "Treated"),
    legend.bty = 1, ylab = "Probability",
    xlab = "Survival Time in Month")

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

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

### Plot Kaplan-Meier curves
plot(survfit(Surv(time, event) ~ group, data = gbm),
     main = "Grade IV Glioma", lty = c(2,1),
     legend.text = c("Control", "Treated"),
     legend.bty = 1, legend.pos = 1, ylab = "Probability",
     xlab = "Survival Time in Month")

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

### stratified logrank test
surv_test(Surv(time, event) ~ group | histology, data = glioma,
          distribution = approximate(B = 10000))

```

hohnloser

Left ventricular ejection fraction of patients with malignant ventricular tachyarrhythmias.

Description

A data frame with the left ventricular ejection fraction of patients with malignant ventricular tachyarrhythmias including recurrence-free month and censoring.

Usage

```
data("hohnloser")
```

Format

EF left ventricular ejection in percent

time recurrence-free month

event censoring: 0 censored, 1 not censored

The data used here is published in Table 1 of Lausen and Schumacher (1992).

Source

The data was first published by Hohnloser et al. (1987), the data used here is published in Table 1 of Lausen and Schumacher (1992).

References

S. H. Hohnloser, E. A. Raeder, P. J. Podrid, T. B. Graboys & B. and Lown, (1987). Predictors of antiarrhythmic drug efficacy in patients with malignant ventricular tachyarrhythmias. *American Heart Journal* **114**, 1–7.

Berthold Lausen & Martin Schumacher (1992). Maximally selected rank statistics. *Biometrics* **48**, 73–85.

Examples

```
maxstat_test(Surv(time, event) ~ EF, data = hohnloser)
```

```
IndependenceLinearStatistic-class
      Class "IndependenceLinearStatistic"
```

Description

Representing linear statistics

Objects from the Class

Objects can be created by calls of the form `new("IndependenceLinearStatistic", ip, xtrafo, ytrafo, ...)`.

Slots

```
linearstatistic: Object of class "numeric" ~~
expectation: Object of class "numeric" ~~
covariance: Object of class "VarCovar" ~~
xtrans: Object of class "matrix" ~~
ytrans: Object of class "matrix" ~~
xtrafo: Object of class "function" ~~
ytrafo: Object of class "function" ~~
x: Object of class "data.frame" ~~
y: Object of class "data.frame" ~~
weights: Object of class "numeric" ~~
block: Object of class "factor" ~~
```

Extends

Class "[IndependenceTestProblem](#)", directly. Class "[IndependenceProblem](#)", by class "[IndependenceTestProblem](#)", distance 2.

Methods

No methods defined with class "IndependenceLinearStatistic" in the signature.

IndependenceProblem-class
Class "IndependenceProblem"

Description

The main class for data handling.

Objects from the Class

Objects can be created by calls of the form `new("IndependenceProblem", x, y, block, weights)`.

Slots

x: Object of class "data.frame".

y: Object of class "data.frame".

weights: Object of class "numeric", however, integer values greater or equal zero are expected (and checked).

block: Object of class "factor".

Methods

initialize signature(.Object = "IndependenceProblem"):...

IndependenceTest *General Independence Tests*

Description

The independence between two sets of variables of arbitrary measurement scales, possibly stratified in blocks, is tested conditional on the data.

Usage

```
## S3 method for class 'formula':
independence_test(formula, data, subset = NULL,
  weights = NULL, ...)
## S3 method for class 'IndependenceProblem':
independence_test(object,
  teststat = c("max", "quad", "scalar"),
  distribution = c("asymptotic", "approximate", "exact"),
  alternative = c("two.sided", "less", "greater"),
  xtrafo = trafo, ytrafo = trafo, scores = NULL,
  check = NULL, ...)
## S3 method for class 'table':
independence_test(object,
  distribution = c("asymptotic", "approximate"), ...)
```

Arguments

<code>formula</code>	a formula of the form $y_1 + \dots + y_p \sim x_1 + \dots + x_q$ <code>block</code> where the variables on the left and right hand side may be measured on arbitrary scales (including censored ones on the left hand side) and <code>block</code> is an optional factor for stratification.
<code>data</code>	an optional data frame containing the variables in the model formula. Alternatively, an object of class <code>exprSet</code> may be specified. In this case, all variables in <code>formula</code> , except <code>.</code> , are first evaluated in the <code>pData</code> data frame. The dot (<code>.</code>) refers to the matrix of expression levels (<code>exprs</code> slot).
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>weights</code>	an optional formula of the form $\sim w$ defining integer valued weights for the observations.
<code>object</code>	an object inheriting from class <code>IndependenceProblem</code> or an object of class <code>table</code> .
<code>teststat</code>	a character, the type of test statistic to be applied: either a standardized scalar test statistic (<code>scalar</code>), or a maximum type statistic (<code>max</code>) or a quadratic form (<code>quad</code>).
<code>alternative</code>	a character, the alternative hypothesis must be one of <code>"two.sided"</code> (default), <code>"greater"</code> or <code>"less"</code> . You can specify just the initial letter.
<code>distribution</code>	a character, the null distribution of the test statistic can be computed exactly or can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>exact</code> , <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated. It is also possible to specify a function with one argument (taking objects inheriting from <code>IndependenceTestStatistic</code>) which return an object of class <code>NullDistribution</code> .
<code>xtrafo</code>	a function of transformations (see <code>trafo</code>) to be applied to the variables on the right hand side of <code>formula</code> , see below.

<code>ytrafo</code>	a function of transformations (see <code>trafo</code>) to be applied to the variables on the left hand side of <code>formula</code> , see below.
<code>scores</code>	a named list of scores to be attached to ordered factors. In case a variable is an unordered factor, it is coerced to <code>ordered</code> first.
<code>check</code>	a function to be applied to objects of class <code>IndependenceTest</code> in order to check for specific properties of the data.
<code>...</code>	further arguments to be passed to or from methods. Currently, none of the additional arguments is passed to any function.

Details

The null hypothesis of the independence between the variables on the left hand side and the variables on the right hand side of `formula`, possibly stratified by `block`, is tested. The vector supplied via the `weights` argument is interpreted as observation counts.

This function is the basic workhorse called by all other convenience functions, mainly by supplying transformations via the `xtrafo` argument and influence functions via the `ytrafo` argument.

The `scores` argument leads to linear-by-linear association tests against ordered alternatives. If the formula `y ~ x` was supplied and both `y` and `x` are factors, `scores = list(y = 1:k, x = c(1, 4, 6))` first triggers a coercion to class `ordered` of both variables and attaches the list elements as scores. The length of a score vector needs to be equal the number of levels of the factor of interest.

The basis of this function is the framework for conditional inference procedures by Strasser & Weber (1999). The theory and this implementation are explained and illustrated in Hothorn, Hornik, van de Wiel and Zeileis (2006).

Value

An object inheriting from class `IndependenceTest-class` with methods `show`, `statistic`, `expectation`, `covariance` and `pvalue`. The null distribution can be inspected by `pperm`, `dperm`, `qperm` and `support` methods.

References

Helmut Strasser & Christian Weber (1999). On the asymptotic theory of permutation statistics. *Mathematical Methods of Statistics* **8**, 220–250.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2006). A Lego System for Conditional Inference. *The American Statistician*, **60**(3), 257–263.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2008). Implementing a class of permutation tests: The coin package, *Journal of Statistical Software*, **28**(8), 1–23. <http://www.jstatsoft.org/v28/i08/>

Examples

```
### independence of asat and group via 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 group
xtrafo = function(data) trafo(data, factor_trafo = function(x)
  matrix(x == levels(x)[1], ncol = 1))
)

### same as
normal_test(asat ~ group, data = asat, distribution = "exact",
  alternative = "greater")

### if you are interested in the internals:
## Not run:
  browseURL(system.file("documentation", "html", "index.html",
    package = "coin"))

## End(Not run)

```

```

IndependenceTest-class
      Class "IndependenceTest"

```

Description

Objects of class "IndependenceTest" represent the results of independence tests, including data, transformations, linear and test statistics as well as their null distribution.

Objects from the Class

Objects can be created by calls of the form `new("IndependenceTest", ...)`. Normally, this class is only used internally.

Slots

distribution: Object of class "NullDistribution".
statistic: Object of class "IndependenceTestStatistic".
estimates: Object of class "list".
method: Object of class "character".

Methods

initialize signature(.Object = "IndependenceTest"):...

```
IndependenceTestProblem-class
      Class "IndependenceTestProblem"
```

Description

Representing an independence problem

Objects from the Class

Objects can be created by calls of the form `new ("IndependenceTestProblem", ip, xtrafo, ytrafo, ...)`.

Slots

xtrans: transformed x values
ytrans: transformed y values
xtrafo: function to transform x values with
ytrafo: function to transform y values with
x: x values, an object of class "data.frame"
y: y values, an object of class "data.frame"
weights: numeric vector of weights
block: factor representing blocks

Extends

Class "[IndependenceProblem](#)", directly.

Methods

initialize signature (.Object = "IndependenceTestProblem"):...

```
IndependenceTestStatistic-class
      Class "IndependenceTestStatistic"
```

Description

Representing test statistics

Objects from the Class

Objects can be created by calls of the form `new ("IndependenceTestStatistic", itp, varonly)`.

Slots

teststatistic: Object of class "numeric" ~~
standardizedlinearstatistic: Object of class "numeric" ~~
linearstatistic: Object of class "numeric" ~~
expectation: Object of class "numeric" ~~
covariance: Object of class "VarCovar" ~~
xtrans: Object of class "matrix" ~~
ytrans: Object of class "matrix" ~~
xtrafo: Object of class "function" ~~
ytrafo: Object of class "function" ~~
x: Object of class "data.frame" ~~
y: Object of class "data.frame" ~~
weights: Object of class "numeric" ~~
block: Object of class "factor" ~~

Extends

Class "[IndependenceLinearStatistic](#)", directly. Class "[IndependenceTestProblem](#)",
 by class "IndependenceLinearStatistic", distance 2. Class "[IndependenceProblem](#)", by class
 "IndependenceLinearStatistic", distance 3.

initialize-methods *Methods for Function initialize in Package 'coin'*

Description

Methods for function initialize in package **coin**

Methods

.Object = "ANY" ~~describe this method here
.Object = "traceable" ~~describe this method here
.Object = "signature" ~~describe this method here
.Object = "environment" ~~describe this method here
.Object = "ModelEnv" ~~describe this method here
.Object = "IndependenceProblem" ~~describe this method here
.Object = "IndependenceTestProblem" ~~describe this method here
.Object = "IndependenceTestStatistic" ~~describe this method here
.Object = "IndependenceLinearStatistic" ~~describe this method here
.Object = "ScalarIndependenceTestStatistic" ~~describe this method here
.Object = "MaxTypeIndependenceTestStatistic" ~~describe this method here
.Object = "QuadTypeIndependenceTestStatistic" ~~describe this method here
.Object = "SymmetryProblem" ~~describe this method here

jobsatisfaction *Income and Job Satisfaction*

Description

Income and job satisfaction by gender.

Usage

```
data("jobsatisfaction")
```

Format

A tree-dimensional contingency table with dimensions

Income a factor at levels <5000, 5000–15000, 15000–25000 and >25000.

Job.Satisfaction a factor at levels Very Dissatisfied, A Little Satisfied, Moderately Satisfied and Very Satisfied.

Gender a factor at levels Female and Male.

Details

The data are given in Table 7.8, page 288, in Agresti (2002).

Source

Alan Agresti (2002), *Categorical Data Analysis*. Hoboken, New Jersey: John Wiley & Sons.

Examples

```
### Generalized Cochran-Mantel-Haenzel test
cmh_test(jobsatisfaction)
```

LocationTests *Independent Two- and K-Sample Location Tests*

Description

Testing the equality of the distributions of a numeric response 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':
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,
             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,
             distribution = c("asymptotic", "approximate"), ...)
```

Arguments

<code>formula</code>	a formula of the form $y \sim x \mid \text{block}$ where y is a numeric variable giving the data values and x a factor with two or more levels giving the corresponding groups. <code>block</code> is an optional factor for stratification.
<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>weights</code>	an optional formula of the form $\sim w$ defining integer valued weights for the observations.
<code>object</code>	an object of class <code>IndependenceProblem</code> .
<code>distribution</code>	a character, the null distribution of the test statistic can be computed exactly or can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>exact</code> ,

	<code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated.
<code>ties.method</code>	a character, two methods are available to adjust scores for ties, either the score generating function is applied to <code>mid-ranks</code> or the scores computed based on random ranks are averaged for all tied values (<code>average-scores</code>).
<code>conf.int</code>	a logical indicating whether a confidence interval for the difference in location should be computed.
<code>conf.level</code>	confidence level of the interval.
<code>...</code>	further arguments to be passed to or from methods.

Details

The null hypothesis of the equality of the distribution of y in the groups given by x is tested. In particular, the methods documented here are designed to detect shift alternatives. For a general description of the test procedures documented here we refer to Hollander & Wolfe (1999).

The test procedures apply a rank transformation to the response values y , except of `oneway_test` which computes a test statistic using the untransformed response values.

The asymptotic null distribution is computed by default for all procedures. Exact p-values may be computed for the two-sample problems and can be approximated via Monte-Carlo resampling for all procedures. Exact p-values are computed either by the shift algorithm (Streitberg & Röhmel, 1986, 1987) or by the split-up algorithm (van de Wiel, 2001).

The linear rank tests for two samples (`wilcox_test`, `normal_test` and `median_test`) can be used to test the two-sided hypothesis $H_0 : Y_1 - Y_2 = 0$, where Y_i is the median of the responses in the i th group. Confidence intervals for the difference in location are available for the rank-based procedures and are computed according to Bauer (1972). In case `alternative = "less"`, the null hypothesis $H_0 : Y_1 - Y_2 \geq 0$ is tested and `alternative = "greater"` corresponds to a null hypothesis $H_0 : Y_1 - Y_2 \leq 0$.

In case x is an ordered factor, `kruskal_test` computes the linear-by-linear association test for ordered alternatives.

For the adjustment of scores for tied values see Hajek, Sidak and Sen (1999), page 131ff.

Value

An object inheriting from class `IndependenceTest-class` with methods `show`, `statistic`, `expectation`, `covariance` and `pvalue`. The null distribution can be inspected by `pperm`, `dperm`, `qperm` and `support` methods. Confidence intervals can be extracted by `confint`.

References

- Myles Hollander & Douglas A. Wolfe (1999). *Nonparametric Statistical Methods, 2nd Edition*. New York: John Wiley & Sons.
- Bernd Streitberg & Joachim Röhmel (1986). Exact distributions for permutations and rank tests: An introduction to some recently published algorithms. *Statistical Software Newsletter* **12**(1), 10–17.
- Bernd Streitberg & Joachim Röhmel (1987). Exakte Verteilungen für Rang- und Randomisierungstests im allgemeinen c -Stichprobenfall. *EDV in Medizin und Biologie* **18**(1), 12–19.

Mark A. van de Wiel (2001). The split-up algorithm: a fast symbolic method for computing p-values of rank statistics. *Computational Statistics* **16**, 519–538.

David F. Bauer (1972). Constructing confidence sets using rank statistics. *Journal of the American Statistical Association* **67**, 687–690.

Jaroslav Hajek, Zbynek Sidak & Pranab K. Sen (1999), *Theory of Rank Tests*. San Diego, London: Academic Press.

Examples

```
### Tritiated Water Diffusion Across Human Chorioamnion
### Hollander & Wolfe (1999), Table 4.1, page 110
water_transfer <- 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(c(rep("At term", 10), rep("12-26 Weeks", 5))))

### Wilcoxon-Mann-Whitney test, cf. Hollander & Wolfe (1999), page 111
### exact p-value and confidence interval for the difference in location
### (At term - 12-26 Weeks)
wt <- wilcox_test(pd ~ age, data = water_transfer,
                 distribution = "exact", conf.int = TRUE)

print(wt)

### extract observed Wilcoxon statistic, i.e, the sum of the
### ranks for age = "12-26 Weeks"
statistic(wt, "linear")

### its expectation
expectation(wt)

### and variance
covariance(wt)

### and, finally, the exact two-sided p-value
pvalue(wt)

### Confidence interval for difference (12-26 Weeks - At term)
wilcox_test(pd ~ age, data = water_transfer,
           xtrafo = function(data)
             trafo(data, factor_trafo = function(x)
                 as.numeric(x == levels(x)[2])),
           distribution = "exact", conf.int = TRUE)

### Permutation test, asymptotic p-value
oneway_test(pd ~ age, data = water_transfer)

### approximate p-value (with 99% confidence interval)
pvalue(oneway_test(pd ~ age, data = water_transfer,
                 distribution = approximate(B = 9999)))

### exact p-value
pt <- oneway_test(pd ~ age, data = water_transfer, distribution = "exact")
```

```

pvalue(pt)

### plot density and distribution of the standardized
### test statistic
layout(matrix(1:2, nrow = 2))
s <- support(pt)
d <- sapply(s, function(x) dperm(pt, x))
p <- sapply(s, function(x) pperm(pt, x))
plot(s, d, type = "S", xlab = "Teststatistic", ylab = "Density")
plot(s, p, type = "S", xlab = "Teststatistic", ylab = "Cumm. Probability")

### Length of YOY Gizzard Shad from Kokosing Lake, Ohio,
### sampled in Summer 1984, Hollander & Wolfe (1999), Table 6.3, page 200
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 = factor(c(rep("I", 10), rep("II", 10),
                                  rep("III", 10), rep("IV", 10))))

### Kruskal-Wallis test, approximate exact p-value
kw <- kruskal_test(length ~ site, data = YOY,
                   distribution = approximate(B = 9999))
kw
pvalue(kw)

### Nemenyi-Damico-Wolfe-Dunn test (joint ranking)
### Hollander & Wolfe (1999), page 244
### (where Steel-Dwass results are given)
if (require("multcomp")) {
  NDWD <- oneway_test(length ~ site, data = YOY,
                     ytrafo = function(data) trafo(data, numeric_trafo = rank),
                     xtrafo = function(data) trafo(data, factor_trafo = function(x)
                                                    model.matrix(~x - 1) %*% t(contrMat(table(x), "Tukey"))),
                     teststat = "max", distribution = approximate(B = 90000))

  ### global p-value
  print(pvalue(NDWD))

  ### sites (I = II) != (III = IV) at alpha = 0.01 (page 244)
  print(pvalue(NDWD, method = "single-step"))
}

```

MarginalHomogeneityTest

Marginal Homogeneity Test

Description

Testing marginal homogeneity in a complete block design.

Usage

```
## S3 method for class 'formula':
mh_test(formula, data, subset = NULL, ...)
## S3 method for class 'table':
mh_test(object, ...)
## S3 method for class 'SymmetryProblem':
mh_test(object, distribution = c("asymptotic", "approximate"), ...)
```

Arguments

<code>formula</code>	a formula of the form $y \sim x \mid \text{block}$ where y is a factor giving the data values and x a factor with two or more levels giving the corresponding replications. <code>block</code> is an optional factor (which is generated automatically when omitted).
<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>object</code>	an object inheriting from class <code>SymmetryProblem</code> or a table with identical <code>dimnames</code> attributes.
<code>distribution</code>	a character, the null distribution of the test statistic can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated.
<code>...</code>	further arguments to be passed to or from methods.

Details

The null hypothesis of independence of row and column totals is tested. The corresponding test for binary factors x and y is known as McNemar test. For larger tables, Stuart's W_0 statistic (Stuart, 1955, Agresti, 2002, page 422, also known as Stuart-Maxwell test) is computed. The marginal homogeneity statistic W of Bhapkar (1966) can be derived from W_0 via $W = W_0/(1 - W_0/n)$ (see Agresti, 2002, page 422).

Scores must be a list of length one (row and column scores coincide). When scores are given or if x is ordered, the corresponding linear association test is computed (see Agresti, 2002).

Note that for a large number of observations, this function is rather inefficient.

Value

An object inheriting from class `IndependenceTest` with methods `show`, `pvalue` and `statistic`.

References

- Alan Agresti (2002). *Categorical Data Analysis*. Hoboken, New Jersey: John Wiley & Sons.
- V. P. Bhapkar (1966). A note on the equivalence of two test criteria for hypotheses in categorical data. *Journal of the American Statistical Association* **61**, 228–235.
- Alan Stuart (1955). A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika* **42**(3/4), 412–416.

Examples

```
### Opinions on Pre- and Extramarital Sex, Agresti (2002), page 421
opinions <- c("always wrong", "almost always wrong",
             "wrong only sometimes", "not wrong at all")

PreExSex <- as.table(matrix(c(144, 33, 84, 126,
                             2, 4, 14, 29,
                             0, 2, 6, 25,
                             0, 0, 1, 5), nrow = 4,
                             dimnames = list(PremaritalSex = opinions,
                                                ExtramaritalSex = opinions)))

### treating response as nominal
mh_test(PreExSex)

### and as ordinal
mh_test(PreExSex, scores = list(response = 1:length(opinions)))

### example taken from
### http://ourworld.compuserve.com/homepages/jsuebersax/mcnemar.htm
rating <- c("low", "moderate", "high")
x <- as.table(matrix(c(20, 10, 5,
                      3, 30, 15,
                      0, 5, 40),
                      ncol = 3, byrow = TRUE,
                      dimnames = list(Rater1 = rating, Rater2 = rating)))
### test statistic W_0 = 13.76
mh_test(x)
```

Description

Testing the independence of a set of ordered or numeric covariates and a response of arbitrary measurement scale against cutpoint alternatives.

Usage

```
## S3 method for class 'formula':
maxstat_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem':
maxstat_test(object,
  distribution = c("asymptotic", "approximate"),
  teststat = c("max", "quad"),
  minprob = 0.1, maxprob = 1 - minprob, ...)
```

Arguments

<code>formula</code>	a formula of the form $y \sim x_1 + \dots + x_p \mid \text{block}$ where y and covariates x_1 to x_p can be variables measured at arbitrary scales; <code>block</code> is an optional factor for stratification.
<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>weights</code>	an optional formula of the form $\sim w$ defining integer valued weights for the observations.
<code>object</code>	an object inheriting from class <code>IndependenceProblem</code> .
<code>distribution</code>	a character, the null distribution of the test statistic can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated.
<code>teststat</code>	a character, the type of test statistic to be applied: a maximum type statistic (<code>max</code>) or a quadratic form (<code>quad</code>).
<code>minprob</code>	a fraction between 0 and 0.5; consider only cutpoints greater than the <code>minprob</code> * 100 % quantile of x .
<code>maxprob</code>	a fraction between 0.5 and 1; consider only cutpoints smaller than the <code>maxprob</code> * 100 % quantile of x .
<code>...</code>	further arguments to be passed to or from methods.

Details

The null hypothesis of independence of all covariates to the response y against simple cutpoint alternatives is tested.

For an unordered covariate x , all possible partitions into two groups are evaluated. The cutpoint is then a set of levels defining one of the two groups.

Value

An object inheriting from class `IndependenceTest-class` with methods `show`, `statistic`, `expectation`, `covariance` and `pvalue`. The null distribution can be inspected by `pperm`, `dperm`, `qperm` and `support` methods.

References

- Rupert Miller & David Siegmund (1982). Maximally Selected Chi Square Statistics. *Biometrics* **38**, 1011–1016.
- Berthold Lausen & Martin Schumacher (1992). Maximally Selected Rank Statistics. *Biometrics* **48**, 73–85.
- Torsten Hothorn & Berthold Lausen (2003). On the Exact Distribution of Maximally Selected Rank Statistics. *Computational Statistics & Data Analysis* **43**, 121–137.
- Berthold Lausen, Torsten Hothorn, Frank Bretz & Martin Schumacher (2004). Optimally Selected Prognostic Factors. *Biometrical Journal* **46**, 364–374.
- Jörg Müller & Torsten Hothorn (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.
- Torsten Hothorn & Achim Zeileis (2008). Generalized maximally selected statistics, *Biometrics*, **64**(4), 1263–1269.

Examples

```
### analysis of the tree pipit data in Mueller and Hothorn (2004)
maxstat_test(counts ~ coverstorey, data = treepipit)

### and for all possible covariates (simultaneously)
mt <- maxstat_test(counts ~ ., data = treepipit)
show(mt)$estimate

### reproduce applications in Sections 7.2 and 7.3
### of Hothorn & Lausen (2003) with limiting distribution

maxstat_test(Surv(time, event) ~ EF, data = hohnloser,
  ytrafo = function(data) trafo(data, surv_trafo = function(x)
    logrank_trafo(x, ties = "HL"))

maxstat_test(Surv(RFS, event) ~ SPF, data = sphase,
  ytrafo = function(data) trafo(data, surv_trafo = function(x)
    logrank_trafo(x, ties = "HL"))
```

MaxTypeIndependenceTestStatistic-class

Class "MaxTypeIndependenceTestStatistic" ~~~

Description

Represent a maximum-type test statistic

Objects from the Class

Objects can be created by calls of the form `new ("MaxTypeIndependenceTestStatistic", its, alternative)`.

Slots

alternative: Object of class "character" ~~
teststatistic: Object of class "numeric" ~~
standardizedlinearstatistic: Object of class "numeric" ~~
linearstatistic: Object of class "numeric" ~~
expectation: Object of class "numeric" ~~
covariance: Object of class "VarCovar" ~~
xtrans: Object of class "matrix" ~~
ytrans: Object of class "matrix" ~~
xtrafo: Object of class "function" ~~
ytrafo: Object of class "function" ~~
x: Object of class "data.frame" ~~
y: Object of class "data.frame" ~~
weights: Object of class "numeric" ~~
block: Object of class "factor" ~~

Extends

Class "[IndependenceTestStatistic](#)", directly. Class "[IndependenceLinearStatistic](#)", by class "IndependenceTestStatistic", distance 2. Class "[IndependenceTestProblem](#)", by class "IndependenceTestStatistic", distance 3. Class "[IndependenceProblem](#)", by class "IndependenceTestStatistic", distance 4.

Methods

initialize signature(.Object = "MaxTypeIndependenceTestStatistic"):...

mercuryfish

Chromosomal Effects of Mercury Contaminated Fish Consumption

Description

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

Usage

```
data("mercuryfish")
```

Format

A data frame with 39 observations on the following 4 variables.

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

mercury the level of mercury in the blood.

abnormal the proportion of cells with structural abnormalities.

ccells the proportion of cells with asymmetrical or incomplete-symmetrical chromosome aberrations called C_u cells.

Details

Subjects who ate contaminated fish for more than three years in the `exposed` group and subjects of a control group are to be compared. Instead of a multivariate comparison, Rosenbaum (1994) applied a coherence criterion. The observations are partially ordered: an observation is smaller than another when all three variables (`mercury`, `abnormal` and `ccells`) are smaller and a score reflecting the ‘ranking’ is attached to each observation. The distribution of the scores in both groups is to be compared and the corresponding test is called ‘POSET-test’ (partially ordered sets).

Source

S. Skerfving, K. Hansson, C. Mangs, J. Lindsten, N. Ryman (1974), Methylmercury-induced chromosome damage in men. *Environmental Research* **7**, 83–98.

References

P. R. Rosenbaum (1994). Coherence in observational studies. *Biometrics* **50**, 368–374.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2006). A Lego system for conditional inference, *The American Statistician*, **60**(3), 257–263.

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)
}

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

### linear statistic (T in Rosenbaum's, 1994, notation)
statistic(poset, "linear")

### expectation
expectation(poset)
```

```

### variance (there is a typo in Rosenbaum, 1994, page 371,
### last paragraph Section 2)
covariance(poset)

### the standardized statistic
statistic(poset)

### and asymptotic p-value
pvalue(poset)

### exact p-value
independence_test(mercury + abnormal + ccells ~ group, data =
                  mercuryfish, ytrafo = coherence, distribution = "exact")

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

### global p-value
pvalue(mvtest)

### adjusted univariate p-value
pvalue(mvtest, method = "single-step")

```

neuropathy

Acute Painful Diabetic Neuropathy

Description

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

Usage

```
data("neuropathy")
```

Format

A data frame with 58 observations on the following 2 variables.

pain pain scores: $\ln(\text{baseline}/\text{final})$.

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

Details

Data from Table 1 of Conover & Salsburg (1988).

Source

William J. Conover & David S. Salsburg (1988). Locally most powerful tests for detecting treatment effects when only a subset of patients can be expected to "respond" to treatment. *Biometrics* **44**, 189–196.

Examples

```
### compare with Table 2 of Conover & Salsburg (1988)
oneway_test(pain ~ group, data = neuropathy, alternative = "less",
            distribution = "exact")

wilcox_test(pain ~ group, data = neuropathy, alternative = "less",
            distribution = "exact")

oneway_test(pain ~ group, data = neuropathy,
            distribution = approximate(B = 10000),
            alternative = "less", ytrafo = function(data) trafo(data,
            numeric_trafo = consal_trafo))
```

```
NullDistribution-class
      Class "NullDistribution"
```

Description

Representing reference distributions

Objects from the Class

Objects can be created by calls of the form `new("NullDistribution", ...)`.

Slots

q: quantile function
d: density function
support: support of the distribution
parameters: list of additional parameters
pvalue: p value function
p: distribution function
name: character, name of the distribution

Extends

Class "[PValue](#)", directly.

`ocarcinoma`*Ovarian Carcinoma*

Description

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

Usage

```
data("ocarcinoma")
```

Format

A data frame with 35 observations on the following 3 variables.

time time in days.

event censoring indicator: FALSE means censored, TRUE is an event.

stadium a factor with levels II and IIA.

Details

Data from Fleming et al. (1980, 1984), reanalysed in Schumacher and Schulgen (2002).

Source

Thomas R. Fleming, Judith R. O'Fallon, Peter C. O'Brien & David P. Harrington (1980). Modified Kolmogorov-Smirnov test procedures with applications to arbitrarily censored data. *Biometrics* **36**, 607–625.

Thomas R. Fleming, Stephanie J. Green & David P. Harrington (1984). Considerations of monitoring and evaluating treatment effects in clinical trials. *Controlled Clinical Trials* **5**, 55–66.

References

Martin Schumacher & Gabi Schulgen (2002), *Methodik klinischer Studien: Methodische Grundlagen der Planung, Durchführung und Auswertung*. Springer, Heidelberg.

Examples

```
### logrank test with exact two-sided p-value
lrt <- surv_test(Surv(time, event) ~ stadium, data = ocarcinoma,
                 distribution = "exact")

### the test statistic
statistic(lrt)

### p-value
pvalue(lrt)
```

 PermutationDistribution

Permutation Distribution of Conditional Independence Tests

Description

Density, distribution, quantile and support of permutation distributions of conditional independence tests.

Usage

```
dperm(object, x, ...)
pperm(object, q, ...)
qperm(object, p, ...)
support(object, ...)
```

Arguments

object	an object inheriting from class <code>IndependenceTest-class</code> .
x	vector of a standardized statistics.
q	vector of a standardized statistics.
p	vector of probabilities.
...	further arguments to be passed to methods.

Methods

dperm density function of a permutation distribution.
pperm distribution function of a permutation distribution.
qperm quantile function for a permutation distribution.
support the support of a permutation distribution.

Examples

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

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

### density of the exact distribution of the Ansari-Bradley statistic
dens <- sapply(support(at), dperm, object = at)

### plot density
plot(support(at), dens, type = "s")
```

```
### 95% quantile
qperm(at, 0.95)

### one-sided p-value
pperm(at, statistic(at))
```

photocar

Multiple Dosing Photocarcinogenicity Experiment

Description

Survival time, time to first tumor and total number of tumors for three groups of animals from a photocarcinogenicity study.

Usage

```
data("photocar")
```

Format

A data frame with 108 observations on the following 6 variables.

group a factor with levels A, B, and C

ntumor total number of tumors.

time survival time.

event censoring indicator (TRUE when the animal died).

dmin time to first tumor.

tumor censoring indicator for dmin, i.e., TRUE when at least one tumor was observed.

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 1-3 in Molefe et al. (2005).

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

Source

Daniel F. Molefe, James J. Chen, Paul C. Howard, Barbara J. Miller, Christopher P. Sambuco, P. Donald Forbes & Ralph L. Kodell (2005). Tests for effects on tumor frequency and latency in multiple dosing photocarcinogenicity experiments. *Journal of Statistical Planning and Inference* **129**, 39–58.

Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel & Achim Zeileis (2006). A Lego system for conditional inference, *The American Statistician*, **60**(3), 257–263.

Examples

```
layout(matrix(1:3, ncol = 3))
plot(survfit(Surv(time, event) ~ group, data = photocar), xmax = 50,
     lty = 1:3, main = "Survival Time")
legend("bottomleft", lty = 1:3, levels(photocar$group), bty = "n")
plot(survfit(Surv(dmin, tumor) ~ group, data = photocar), xmax = 50,
     lty = 1:3, main = "Time to First Tumor")
legend("bottomleft", lty = 1:3, levels(photocar$group), bty = "n")
boxplot(ntumor ~ group, data = photocar, main = "Number of Tumors")

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

pvalue(it)

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

PValue-class

Class "PValue"

Description

A class for p value functions

Objects from the Class

Objects can be created by calls of the form `new("PValue", ...)`.

Slots

pvalue: p value function

p: distribution function

name: character, name of the distribution

Methods

No methods defined with class "PValue" in the signature.

pvalue-methods *Extract P-Values*

Description

Extracts the p-value from objects representing null distributions of independence tests.

Usage

```
pvalue(object, ...)
```

Arguments

object	an object inheriting from class <code>IndependenceTest-class</code> .
...	additional arguments: method, a character specifying the type of adjustment (global, single-step, step-down or discrete) should be used. The default is global.

Details

Univariate p-values for maximum-type statistics come with associated 99% confidence interval when resampling was used to determine the null distribution (which may be the case even when `distribution = "asypmtotic"` was used).

By default, a global p-value is returned. When `method = "single-step"`, adjusted p-values are obtained from a single-step max-T procedure (Westfall & Young, 1993, algorithm 2.5 and formula 2.8). Note that the minimum of the adjusted p-values always controls the familywise error rate (FWER) but the maximum type I error, i.e. the error for each of the individual tests, is only controlled when the subset pivotality condition holds.

When `method = "step-down"` the free step-down resampling method (algorithm 2.8 and formula 2.8 in Westfall & Young, 1993) is used, the above comments apply as well.

With `method = "discrete"`, the Bonferroni adjustment as suggested by Westfall & Wolfinger (1997) with improvements for highly discrete permutation distributions is available, however, without taking correlations between the test statistics into account. Here, the p-values are valid even without assuming subset pivotality.

Methods

pvalue extracts the p-value from the specified object.

References

Peter H. Westfall & S. Stanley Young (1993). *Resampling-based Multiple Testing*. New York: John Wiley & Sons.

Peter H. Westfall & Russell D. Wolfinger (1997). Multiple tests with discrete distributions. *The American Statistician* **51**, 3–8.

Examples

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

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

pvalue(at)

### bivariate 2-sample problem
df <- data.frame(y1 = rnorm(20) + c(rep(0, 10), rep(1, 10)),
                 y2 = rnorm(20),
                 x = gl(2, 10))

it <- independence_test(y1 + y2 ~ x, data = df,
                        distribution = approximate(B = 9999))
pvalue(it, method = "single-step")
pvalue(it, method = "step-down")
pvalue(it, method = "discrete")
```

QuadTypeIndependenceTestStatistic-class

Class "QuadTypeIndependenceTestStatistic" ~~~

Description

Represent quadratic forms

Objects from the Class

Objects can be created by calls of the form `new("QuadTypeIndependenceTestStatistic", its, ...)`.

Slots

covarianceplus: Object of class "matrix" ~~

df: Object of class "numeric" ~~

teststatistic: Object of class "numeric" ~~

standardizedlinearstatistic: Object of class "numeric" ~~
linearstatistic: Object of class "numeric" ~~
expectation: Object of class "numeric" ~~
covariance: Object of class "VarCovar" ~~
xtrans: Object of class "matrix" ~~
ytrans: Object of class "matrix" ~~
xtrafo: Object of class "function" ~~
ytrafo: Object of class "function" ~~
x: Object of class "data.frame" ~~
y: Object of class "data.frame" ~~
weights: Object of class "numeric" ~~
block: Object of class "factor" ~~

Extends

Class "[IndependenceTestStatistic](#)", directly. Class "[IndependenceLinearStatistic](#)", by class "IndependenceTestStatistic", distance 2. Class "[IndependenceTestProblem](#)", by class "IndependenceTestStatistic", distance 3. Class "[IndependenceProblem](#)", by class "IndependenceTestStatistic", distance 4.

Methods

initialize signature(.Object = "QuadTypeIndependenceTestStatistic"):...

 rotarod

Rotating Rats Data

Description

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

Usage

```
data("rotarod")
```

Format

A data frame with 24 observations on the following 2 variables.

time the endurance time

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

Details

The 24 rats received a fixed oral dose of a centrally acting muscle relaxant (`treatment`) or a saline solvent (`control`). They were placed on a rotating cylinder and the length of time each rat remains on the cylinder is measured, up to a maximum of 300 seconds. The rats were randomly assigned to the control and treatment group.

Note that the empirical variance in the control group is 0 and that the group medians are identical.

This dataset serves as the basis of an comparison of the results of the Wilcoxon-Mann-Whitney test computed by 11 statistical packages in Bergmann et al. (2000). The exact conditional p-value is 0.0373 (two-sided) and 0.0186 (one-sided). The asymptotic two-sided p-value (corrected for ties) is reported as 0.0147.

Source

Reinhard Bergmann, John Ludbrook & Will P. J. M. Spooren (2000). Different outcomes of the Wilcoxon-Mann-Whitney test from different statistics packages. *The American Statistician* **54**(1), 72–77.

Examples

```
### Wilcoxon-Mann-Whitney Rank Sum Test

### one-sided exact (0.0186)
wilcox_test(time ~ group, data = rotarod,
             alternative = "greater", distribution = "exact")
### two-sided exact (0.0373)
wilcox_test(time ~ group, data = rotarod, distribution = "exact")
### two-sided asymptotical (0.0147)
wilcox_test(time ~ group, data = rotarod)
```

```
ScalarIndependenceTestStatistic-class
Class "ScalarIndependenceTestStatistic"
```

Description

Represent scalar test statistics

Objects from the Class

Objects can be created by calls of the form `new("ScalarIndependenceTestStatistic", its, alternative)`.

Slots

alternative: Object of class "character" ~~
teststatistic: Object of class "numeric" ~~
standardizedlinearstatistic: Object of class "numeric" ~~
linearstatistic: Object of class "numeric" ~~
expectation: Object of class "numeric" ~~
covariance: Object of class "VarCovar" ~~
xtrans: Object of class "matrix" ~~
ytrans: Object of class "matrix" ~~
xtrafo: Object of class "function" ~~
ytrafo: Object of class "function" ~~
x: Object of class "data.frame" ~~
y: Object of class "data.frame" ~~
weights: Object of class "numeric" ~~
block: Object of class "factor" ~~

Extends

Class "[IndependenceTestStatistic](#)", directly. Class "[IndependenceLinearStatistic](#)", by class "IndependenceTestStatistic", distance 2. Class "[IndependenceTestProblem](#)", by class "IndependenceTestStatistic", distance 3. Class "[IndependenceProblem](#)", by class "IndependenceTestStatistic", distance 4.

Methods

initialize signature(.Object = "ScalarIndependenceTestStatistic"):...

 ScaleTests

Independent Two- and K-Sample Scale Tests

Description

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

Usage

```
## S3 method for class 'formula':
ansari_test(formula, data, subset = NULL, weights = NULL, ...)
## S3 method for class 'IndependenceProblem':
ansari_test(object,
  alternative = c("two.sided", "less", "greater"),
  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"),
  distribution = c("asymptotic", "approximate"),
  ...)
```

Arguments

<code>formula</code>	a formula of the form $y \sim x \mid \text{block}$ where y is a numeric variable giving the data values and x a factor with two or more levels giving the corresponding groups. <code>block</code> is an optional factor for stratification.
<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>weights</code>	an optional formula of the form $\sim w$ defining integer valued weights for the observations.
<code>object</code>	an object of class <code>IndependenceProblem</code> .
<code>alternative</code>	a character, the alternative hypothesis must be one of <code>"two.sided"</code> (default), <code>"greater"</code> or <code>"less"</code> . You can specify just the initial letter.
<code>distribution</code>	a character, the null distribution of the test statistic can be computed exactly or can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>exact</code> , <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated.
<code>ties.method</code>	a character, two methods are available to adjust scores for ties, either the score generating function is applied to <code>mid-ranks</code> or the scores computed based on random ranks are averaged for all tied values (<code>average-scores</code>).
<code>conf.int</code>	a logical indicating whether a confidence interval for the difference in location should be computed.
<code>conf.level</code>	confidence level of the interval.
<code>...</code>	further arguments to be passed to or from methods.

Details

The null hypothesis of the equality of the distribution of y in the groups given by x is tested. In particular, the methods documented here are designed to detect scale alternatives. For a general description of the test procedures documented here we refer to Hollander & Wolfe (1999).

The asymptotic null distribution is computed by default for both procedures. Exact p-values may be computed for the Ansari-Bradley test can be approximated via Monte-Carlo for the Fligner-Killeen procedure. Exact p-values are computed either by the shift algorithm (Streitberg & Röhmel, 1986, 1987) or by the split-up algorithm (van de Wiel, 2001).

The Ansari-Bradley test can be used to test the two-sided hypothesis $var(Y_1)/var(Y_2) = 1$, where $var(Y_i)$ is the variance of the responses in the i th group. Confidence intervals for the ratio of scales are available for the Ansari-Bradley test and are computed according to Bauer (1972). In case `alternative = "less"`, the null hypothesis $var(Y_1)/var(Y_2) \geq 1$ is tested and `alternative = "greater"` corresponds to $var(Y_1)/var(Y_2) \leq 1$.

For the adjustment of scores for tied values see Hajek, Sidak and Sen (1999), page 131ff.

Value

An object inheriting from class `IndependenceTest-class` with methods `show`, `statistic`, `expectation`, `covariance` and `pvalue`. The null distribution can be inspected by `pperm`, `dperm`, `qperm` and `support` methods. Confidence intervals can be extracted by `confint`.

References

- Myles Hollander & Douglas A. Wolfe (1999). *Nonparametric Statistical Methods, 2nd Edition*. New York: John Wiley & Sons.
- Bernd Streitberg & Joachim Röhmel (1986). Exact distributions for permutations and rank tests: An introduction to some recently published algorithms. *Statistical Software Newsletter* **12**(1), 10–17.
- Bernd Streitberg & Joachim Röhmel (1987). Exakte Verteilungen für Rang- und Randomisierungstests im allgemeinen c -Stichprobenfall. *EDV in Medizin und Biologie* **18**(1), 12–19.
- Mark A. van de Wiel (2001). The split-up algorithm: a fast symbolic method for computing p-values of rank statistics. *Computational Statistics* **16**, 519–538.
- David F. Bauer (1972). Constructing confidence sets using rank statistics. *Journal of the American Statistical Association* **67**, 687–690.
- Jaroslav Hajek, Zbynek Sidak & Pranab K. Sen (1999). *Theory of Rank Tests*. San Diego, London: Academic Press.

Examples

```
### Serum Iron Determination Using Hyland Control Sera
### Hollander & Wolfe (1999), page 147
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 = factor(gl(2, 20), labels = c("Ramsay", "Jung-Parekh")))
```

```

### Ansari-Bradley test, asymptotical p-value
ansari_test(serum ~ method, data = sid)

### exact p-value
ansari_test(serum ~ method, data = sid, distribution = "exact")

### Platelet Counts of Newborn Infants
### Hollander & Wolfe (1999), Table 5.4, page 171
platalet_counts <- data.frame(
  counts = c(120, 124, 215, 90, 67, 95, 190, 180, 135, 399,
            12, 20, 112, 32, 60, 40),
  treatment = factor(c(rep("Prednisone", 10), rep("Control", 6))))

### Lepage test, Hollander & Wolfe (1999), page 172
lt <- independence_test(counts ~ treatment, data = platalet_counts,
  ytrafo = function(data) trafo(data, numeric_trafo = function(x)
    cbind(rank(x), ansari_trafo(x))),
  teststat = "quad", distribution = approximate(B = 9999))

lt

### where did the rejection come from? Use maximum statistic
### instead of a quadratic form
ltmax <- independence_test(counts ~ treatment, data = platalet_counts,
  ytrafo = function(data) trafo(data, numeric_trafo = function(x)
    matrix(c(rank(x), ansari_trafo(x)), ncol = 2,
            dimnames = list(1:length(x), c("Location", "Scale")))),
  teststat = "max")

### points to a difference in location
pvalue(ltmax, method = "single-step")

### Funny: We could have used a simple Bonferroni procedure
### since the correlation between the Wilcoxon and Ansari-Bradley
### test statistics is zero
covariance(ltmax)

```

show-methods

Methods for Function show in Package 'coin'

Description

Methods for function show in package **coin**

Methods

object = "IndependenceTest" ~~describe this method here

object = "QuadTypeIndependenceTest" ~~describe this method here

```

object = "MaxTypeIndependenceTest" ~~describe this method here
object = "ScalarIndependenceTest" ~~describe this method here
object = "ScalarIndependenceTestConfint" ~~describe this method here

```

SpearmanTest *Spearman's Test on Independence*

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"), ...)

```

Arguments

<code>formula</code>	a formula of the form $y \sim x \mid \text{block}$ where y and x are numeric variables and <code>block</code> is an optional factor for stratification.
<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>weights</code>	an optional formula of the form $\sim w$ defining integer valued weights for the observations.
<code>object</code>	an object of class <code>IndependenceProblem</code> .
<code>distribution</code>	a character, the null distribution of the test statistic can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be approximated.
<code>...</code>	further arguments to be passed to or from methods.

Details

The null hypothesis of the independence of y and x is tested.

Value

An object inheriting from class `IndependenceTest-class` with methods `show`, `statistic`, `expectation`, `covariance` and `pvalue`. The null distribution can be inspected by `pperm`, `dperm`, `qperm` and `support` methods.

Examples

```
spearman_test(CONT ~ INTG, data = USJudgeRatings)
```

sphase

S-phase Fraction of Tumor Cells

Description

S-phase fraction of tumor cells in breast cancer patients.

Usage

```
data("sphase")
```

Format

This data frame contains the following columns:

SPF S-phase fraction

RFS recurrence free survival

event censoring indicator: FALSE means censored, TRUE is an event.

Details

The data have been used to address the question whether a simple cutpoint in S-phase fraction can be used to discriminate between patients with good and bad prognosis (for example in Hothorn & Lausen, 2003).

Source

J. Pfisterer, F. Kommoss, W. Sauerbrei, D. Menzel, M. Kiechle, E. Giese, M. Hilgarth & A. Pfeiderer (1995). DNA flow cytometry in node positive breast cancer: Prognostic value and correlation to morphological and clinical factors. *Analytical and Quantitative Cytology and Histology* **7**(6), 406–412.

References

Torsten Hothorn & Berthold Lausen (2003). On the exact distribution of maximally selected rank statistics. *Computational Statistics & Data Analysis* **43**, 121–137.

Examples

```
maxstat_test(Surv(RFS, event) ~ SPF, data = sphase)

### reproduce the test statistic reported in Hothorn & Lausen (2003)
maxstat_test(Surv(RFS, event) ~ SPF, data = sphase,
  ytrafo = function(data) trafo(data, surv_trafo = function(x)
    logrank_trafo(x, ties.method = "HL")))
```

statistic-methods *Extract Test Statistics, Linear Statistics and Standardized Statistics*

Description

Extract the test statistic and, possibly multivariate, linear statistics in their raw and standardized form from objects of class [IndependenceTest-class](#).

Usage

```
statistic(object, type = c("test", "linear", "standardized"), ...)
```

Arguments

object	an object inheriting from class IndependenceTest-class .
type	either <code>test</code> for test statistic, <code>linear</code> for the unstandardized linear statistic or <code>standardized</code> for the standardized linear statistic.
...	further arguments (currently ignored).

Methods

statistic extracts the specified statistic from `object`.

Examples

```
df <- data.frame(y = gl(4, 5), x = gl(5, 4))

### Cochran-Mantel-Haenzel Test
ct <- cmh_test(y ~ x, data = df)

### chi-squared statistic
statistic(ct)

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

### the same
```

```
table(df$x, df$y)

### and the standardized contingency table for illustrating
### departures from the null hypothesis of independence of x and y
statistic(ct, type = "standardized")
```

SurvTest

*Independent Two- and K-Sample Tests for Censored Data***Description**

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

Usage

```
## S3 method for class 'formula':
surv_test(formula, data, subset = NULL,
           weights = NULL, ...)
## S3 method for class 'IndependenceProblem':
surv_test(object,
           ties.method = c("logrank", "HL", "average-scores"), ...)
```

Arguments

formula	a formula of the form <code>Surv(time, event) ~ x block</code> where <code>time</code> is a positive numeric variable denoting the survival time and <code>event</code> is a logical being <code>TRUE</code> when the event of interest was observed and <code>FALSE</code> in case of censoring. <code>x</code> is a factor with two or more levels giving the corresponding groups. <code>block</code> 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.
weights	an optional formula of the form <code>~ w</code> defining integer valued weights for the observations.
object	an object of class <code>IndependenceProblem</code> .
ties.method	a character specifying the way ties are handled in the definition of the logrank scores, see below.
...	further arguments to be passed to or from methods.

Details

The null hypothesis of the equality of the distribution of the survival functions in the groups induced by `x` is tested.

The test implemented here is based on the classical logrank test, reformulated as a linear rank test. There are several ways of dealing with ties. Here, three methods are implemented. The first one

(`ties.method = "logrank"`) is described, for example, in Kalbfleisch & Prentice (2002, page 221f) or in Callaert (2003) and lead to coefficients

$$a_i = \delta_i - \sum_{j: X_j \leq X_i} \delta_j / (n - |\{k : X_k < X_j\}|)$$

for a linear rank statistic $T = \sum_{i=1}^n a_i U_i$ (in two-sample situations where $U_i = 0$ or $U_i = 1$ denotes the groups) with survival times $X_{.i}$ and censoring indicator $\delta_i = 0$ for censored observations. For further details, see Kalbfleisch & Prentice (2002). The second method is described in Hothorn & Lausen (2003) where the coefficients

$$a_i = \delta_i - \sum_{j: X_j \leq X_i} \delta_j / (n - |\{k : X_k \leq X_j\}| + 1)$$

are suggested. Finally, average scores (as for example used in StatXact) are offered as well.

Note, however, that the test statistics will differ from the results of `survdiff` since the conditional variance is not identical to the variance estimate used by the classical logrank test.

Value

An object inheriting from class `IndependenceTest-class` with methods `show`, `statistic`, `expectation`, `covariance` and `pvalue`. The null distribution can be inspected by `pperm`, `dperm`, `qperm` and `support` methods.

References

John D. Kalbfleisch & Ross L. Prentice (2002), *The Statistical Analysis of Failure Time Data* (2nd edition). John Wiley & Sons, Hoboken, New Jersey.

Herman Callaert (2003), Comparing Statistical Software Packages: The Case of the Logrank Test in StatXact. *The American Statistician* **57**, 214–217.

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

Examples

```
### asymptotic tests for carcinoma data
surv_test(Surv(time, event) ~ stadium, data = ocarcinoma)
survdiff(Surv(time, event) ~ stadium, data = ocarcinoma)

### example data given in Callaert (2003)
exdata <- data.frame(time = c(1, 1, 5, 6, 6, 6, 6, 2, 2, 2, 3, 4, 4, 5, 5),
                     event = rep(TRUE, 15),
                     group = factor(c(rep(0, 7), rep(1, 8))))

### p = 0.0523
survdiff(Surv(time, event) ~ group, data = exdata)
### p = 0.0505
surv_test(Surv(time, event) ~ group, data = exdata,
          distribution = exact())
### p = 0.0468
```

```

surv_test(Surv(time, event) ~ group, data = exdata,
          distribution = exact(), ties = "average")

### lung cancer example from StatXact
`lungcancer` <- structure(list(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 = structure(c(2L, 2L, 2L, 2L, 2L, 1L, 1L,
    1L, 1L, 1L, 1L, 1L, 1L), .Label = c("control", "newdrug"),
    class = "factor")),
    .Names = c("time", "event", "group"),
    row.names = c("1", "2", "3", "4", "5", "6", "7", "8", "9",
    "10", "11", "12", "13", "14"),
    class = "data.frame")

### StatXact 6 manual, page 414
logrank_trafo(Surv(lungcancer$time, lungcancer$event),
              ties = "average")

### StatXact 6 manual, page 415
surv_test(Surv(time, event) ~ group, data = lungcancer,
          ties = "average", distribution = exact())

```

SymmetryProblem-class

Class "SymmetryProblem"

Description

The main class for data handling for symmetry problems.

Objects from the Class

Objects can be created by calls of the form `new("SymmetryProblem", x, y, block, weights)`.

Slots

x: Object of class "data.frame".

y: Object of class "data.frame".

weights: Object of class "numeric", however, integer values greater or equal zero are expected (and checked).

block: Object of class "factor".

Extends

Class "IndependenceProblem", directly.

Methods

initialize signature(.Object = "SymmetryProblem"):...

SymmetryTest	<i>General Symmetry Test</i>
--------------	------------------------------

Description

Testing the symmetry of a response for repeated measurements in a complete block design.

Usage

```
## S3 method for class 'formula':
symmetry_test(formula, data, subset = NULL, ...)
## S3 method for class 'SymmetryProblem':
symmetry_test(object,
  teststat = c("max", "quad", "scalar"),
  distribution = c("asymptotic", "approximate", "exact"),
  alternative = c("two.sided", "less", "greater"),
  xtrafo = trafo, ytrafo = trafo, scores = NULL,
  check = NULL, ...)
## S3 method for class 'table':
symmetry_test(object, ...)
```

Arguments

formula	a formula of the form $y_1 + \dots + y_p \sim x_1 + \dots + x_q$ block where the variables on the left and right hand side may be measured on arbitrary scales 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.
object	an object inheriting from class <code>SymmetryProblem</code> or an object of class <code>table</code> .
teststat	a character, the type of test statistic to be applied: either a standardized scalar test statistic (<code>scalar</code>), or a maximum type statistic (<code>max</code>) or a quadratic form (<code>quad</code>).
alternative	a character, the alternative hypothesis must be one of <code>"two.sided"</code> (default), <code>"greater"</code> or <code>"less"</code> . You can specify just the initial letter.
distribution	a character, the null distribution of the test statistic can be computed exactly or can be approximated by its asymptotic distribution (<code>asymptotic</code>) or via Monte-Carlo resampling (<code>approximate</code>). Alternatively, the functions <code>exact</code> , <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated.
xtrafo	a function of transformations (see <code>trafo</code>) to be applied to the variables on the right hand side of formula, see below.

<code>ytrafo</code>	a function of transformations (see <code>trafo</code>) to be applied to the variables on the left hand side of <code>formula</code> , see below.
<code>scores</code>	a named list of scores to be attached to ordered factors. In case a variable is an unordered factor, it is coerced to <code>ordered</code> first.
<code>check</code>	a function to be applied to objects of class <code>SymmetryTest</code> in order to check for specific forms, see below.
<code>...</code>	further arguments to be passed to or from methods.

Details

This function handles symmetry problems where each block corresponds to exactly one observation with repeated measurements.

Value

An object inheriting from class `IndependenceTest` with methods `show`, `pvalue` and `statistic`.

SymmetryTests	<i>Symmetry Tests</i>
---------------	-----------------------

Description

Testing the symmetry of a response for repeated measurements in a complete block design.

Usage

```
## S3 method for class 'formula':
friedman_test(formula, data, subset = NULL, ...)
## S3 method for class 'SymmetryProblem':
friedman_test(object, distribution = c("asymptotic", "approximate"),
              ...)

## S3 method for class 'formula':
wilcoxsign_test(formula, data, subset = NULL, ...)
## S3 method for class 'IndependenceProblem':
wilcoxsign_test(object,
                ties.method = c("HollanderWolfe", "Pratt"), ...)
```

Arguments

<code>formula</code>	a formula of the form $y \sim x \mid \text{block}$ where y is a numeric variable giving the data values and x a factor with two (<code>wilcoxsign_test</code>) or more levels giving the corresponding groups. <code>block</code> is an optional factor (which is generated automatically when omitted).
----------------------	--

<code>data</code>	an optional data frame containing the variables in the model formula.
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>object</code>	an object inheriting from class <code>SymmetryProblem</code> .
<code>distribution</code>	a character, the null distribution of the test statistic can be approximated by its asymptotic distribution (<code>asympt</code>) or via Monte-Carlo resampling (<code>approx</code>). Alternatively, the functions <code>approximate</code> or <code>asymptotic</code> can be used to specify how the exact conditional distribution of the test statistic should be calculated or approximated. For the Wilcoxon signed rank test, <code>exact</code> computes the exact distribution by means of the Streitberg-Röhmel shift algorithm.
<code>ties.method</code>	a character specifying the way ties are handled, see below.
<code>...</code>	further arguments to be passed to or from methods.

Details

The null hypothesis of symmetry of y across x is tested. In case x is an ordered factor `friedman_test` performs the Page test, scores can be altered by the `scores` argument (see `symmetry_test`).

For `wilcoxsign_test`, formulae of the form $y \sim x \mid \text{block}$ and $y \sim x$ are allowed. The latter is interpreted in the sense that y is the first and x the second measurement on the same observation. By default, zero differences are discarded first and then rank the remaining absolute differences (as explained in Hollander & Wolfe, 1999). Alternatively, following Pratt (1959), we rank the absolute differences (including zeros) first, then discard the ranks corresponding to zero differences, but keep the other ranks as they are.

Value

An object inheriting from class `IndependenceTest` with methods `show`, `pvalue` and `statistic`.

References

Myles Hollander & Douglas A. Wolfe (1999). *Nonparametric Statistical Methods, 2nd Edition*. New York: John Wiley & Sons.

John W. Pratt (1959). Remarks on zeros and ties in the Wilcoxon signed rank procedures. *Journal of the American Statistical Association*, **54**(287), 655–667.

Bernd Streitberg & Joachim Röhmel (1986). Exact distributions for permutations and rank tests: An introduction to some recently published algorithms. *Statistical Software Newsletter* **12**(1), 10–17.

Bernd Streitberg & Joachim Röhmel (1987). Exakte Verteilungen für Rang- und Randomisierungsstests im allgemeinen c -Stichprobenfall. *EDV in Medizin und Biologie* **18**(1), 12–19.

Examples

```
### Hollander & Wolfe (1999), Table 7.1, page 274
### Comparison of three methods ("round out", "narrow angle", and
### "wide angle") for rounding first base.
RoundingTimes <- 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(c("Round Out", "Narrow Angle", "Wide Angle"), 22)),
  block = factor(rep(1:22, rep(3, 22))))

### classical global test
friedman_test(times ~ methods | block, data = RoundingTimes)

### parallel coordinates plot
matplot(t(matrix(RoundingTimes$times, ncol = 3, byrow = TRUE)),
        type = "l", col = 1, lty = 1, axes = FALSE, ylab = "Time",
        xlim = c(0.5, 3.5))
axis(1, at = 1:3, labels = levels(RoundingTimes$methods))
axis(2)

### where do the differences come from?
### Wilcoxon-Nemenyi-McDonald-Thompson test
### Hollander & Wolfe (1999), page 295
if (require("multcomp")) {

  ### all pairwise comparisons
  rtt <- symmetry_test(times ~ methods | block, data = RoundingTimes,
    teststat = "max",
    xtrafo = function(data)
      trafo(data, factor_trafo = function(x)
        model.matrix(~ x - 1) %*% t(contrMat(table(x), "Tukey"))
      ),
    ytrafo = function(data)
      trafo(data, numeric_trafo = rank, block = RoundingTimes$block)
  )

  ### a global test, again
  print(pvalue(rtt))

  ### simultaneous P-values for all pair comparisons

```

```

    ### Wide Angle vs. Round Out differ (Hollander and Wolfe, 1999, page 296)
    print(pvalue(rtt, method = "single-step"))
  }

  ### Strength Index of Cotton, Hollander & Wolfe (1999), Table 7.5, page 286
  sc <- data.frame(block = factor(c(rep(1, 5), rep(2, 5), rep(3, 5))),
    potash = ordered(rep(c(144, 108, 72, 54, 36), 3)),
    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))

  ### Page test for ordered alternatives
  ft <- friedman_test(strength ~ potash | block, data = sc)
  ft

  ### one-sided p-value
  1 - pnorm(sqrt(statistic(ft)))

  ### approximate null distribution via Monte-Carlo
  pvalue(friedman_test(strength ~ potash | block, data = sc,
    distribution = approximate(B = 9999)))

  ### example from ?wilcox.test
  x <- c(1.83, 0.50, 1.62, 2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
  y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
  wilcoxsign_test(x ~ y, alternative = "greater", distribution = exact())
  wilcox.test(x, y, paired = TRUE, alternative = "greater")

  ### with explicit group and block information
  xydat <- data.frame(y = c(y, x), x = gl(2, length(x)),
    block = factor(rep(1:length(x), 2)))
  wilcoxsign_test(y ~ x | block, data = xydat,
    alternative = "greater", distribution = exact())

```

Transformations *Functions for Data Transformations*

Description

Rank-transformations for numerical data or dummy codings of factors.

Usage

```

trafo(data, numeric_trafo = id_trafo, factor_trafo = f_trafo,
  ordered_trafo = of_trafo, surv_trafo = logrank_trafo,
  var_trafo = NULL, block = NULL)
id_trafo(x)
ansari_trafo(x, ties.method = c("mid-ranks", "average-scores"))
fligner_trafo(x, ties.method = c("mid-ranks", "average-scores"))

```

```

normal_trafo(x, ties.method = c("mid-ranks", "average-scores"))
median_trafo(x)
consal_trafo(x, ties.method = c("mid-ranks", "average-scores"))
maxstat_trafo(x, minprob = 0.1, maxprob = 1 - minprob)
logrank_trafo(x, ties.method = c("logrank", "HL", "average-scores"))
f_trafo(x)
of_trafo(x)

```

Arguments

<code>data</code>	an object of class <code>data.frame</code> .
<code>numeric_trafo</code>	a function to be applied to numeric elements of data returning a matrix with <code>nrow(data)</code> rows and an arbitrary number of columns.
<code>factor_trafo</code>	a function to be applied to factor elements of data returning a matrix with <code>nrow(data)</code> rows and an arbitrary number of columns (usually a dummy or contrast matrix).
<code>ordered_trafo</code>	a function to be applied to ordered elements of data returning a matrix with <code>nrow(data)</code> rows and an arbitrary number of columns (usually some scores).
<code>surv_trafo</code>	a function to be applied to elements of class <code>Surv</code> of data returning a matrix with <code>nrow(data)</code> rows and an arbitrary number of columns.
<code>var_trafo</code>	an optional named list of functions to be applied to the corresponding variables in data.
<code>block</code>	an optional factor whose levels are interpreted as blocks. <code>trafo</code> is applied to each level of <code>block</code> separately.
<code>x</code>	an object of classes <code>numeric</code> , <code>ordered</code> , <code>factor</code> or <code>Surv</code> .
<code>ties.method</code>	two methods are available to adjust scores for ties. Either the score generating function is applied to mid-ranks or scores, based on random ranks, are averaged <code>average-scores</code> . For ties handling in case of censored data, see surv_test .
<code>minprob</code>	a fraction between 0 and 0.5.
<code>maxprob</code>	a fraction between 0.5 and 1.

Details

The utility functions documented here are used to define special independence tests.

`trafo` applies its arguments to the elements of data according to the classes of the elements.

`id_trafo` is the identity transformation and `f_trafo` computes dummy matrices for factors.

`ansari_trafo` and `fligner_trafo` compute Ansari-Bradley or Fligner scores for scale problems.

`normal_trafo`, `median_trafo` and `consal_trafo` implement normal scores, median scores or Conover-Salburg scores (see [neuropathy](#)) for location problems, `logrank_trafo` returns logrank scores for censored data.

A `trafo` function with modified default arguments is usually feeded into `independence_test` via the `xtrafo` or `ytrafo` arguments.

Fine tuning (different transformations for different variables) is possible by supplying a named list of functions to the `var_trafo` argument.

Value

A named matrix with `nrow(data)` rows and arbitrary number of columns. User-supplied transformations must return a numeric vector or matrix.

Examples

```
### dummy matrices, 2-sample problem (only one column)
f_trafo(y <- gl(2, 5))

### score matrices
of_trafo(y <- ordered(gl(3, 5)))

### K-sample problem (K columns)
f_trafo(y <- gl(5, 2))

### normal scores
normal_trafo(x <- rnorm(10))

### and now together
trafo(data.frame(x = x, y = y), numeric_trafo = normal_trafo)

### the same, more flexible when multiple variables are in play
trafo(data.frame(x = x, y = y), var_trafo = list(x = normal_trafo))

### maximally selected statistics
maxstat_trafo(rnorm(10))

### apply transformation blockwise (e.g. for Friedman test)
trafo(data.frame(y = 1:20), numeric_trafo = rank, block = gl(4, 5))
```

treepipit

Tree Pipit (Anthus trivialis) Forest Data

Description

Data on the population density of tree pipits in Franconian oak forests including variables describing the forest ecosystem.

Usage

```
data("treepipit")
```

Format

A data frame with 86 observations describing 86 stands with the following 10 variables:

counts the number of tree pipits observed.

age the ages of the overstorey oaks taken from forest data.

coverstorey percentage of 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 percent.

coverregen percentage of cover of regeneration and shrubs.

meanregen mean height of regeneration and shrubs.

coniferous percentage of coniferous trees.

deadtree number of dead trees per grid.

cbpiles number of crowns and branch piles. All laying crowns and branch piles per hectare were counted. These were induced by logging and the creation of wind breaks.

ivytree number of ivied trees. All ivied trees were counted 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 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 samples from March to June 2002 five times, using a quantitative grid mapping. Each grid was a 1-ha square. In total, 86 sample sites weres established. 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.

References

Jörg Müller & Torsten Hothorn (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**, 219–228.

Examples

```
maxstat_test(counts ~ age + coverstorey + coverregen + meanregen +
             coniferous + deadtree + cbpiles + ivytree,
             data = treepipit)
```

Index

*Topic **classes**

ExpectCovar-class, [11](#)
ExpectCovarInfluence-class,
[11](#)
IndependenceLinearStatistic-class,
[14](#)
IndependenceProblem-class, [15](#)
IndependenceTest-class, [18](#)
IndependenceTestProblem-class,
[19](#)
IndependenceTestStatistic-class,
[19](#)
MaxTypeIndependenceTestStatistic-class,
[29](#)
NullDistribution-class, [33](#)
PValue-class, [37](#)
QuadTypeIndependenceTestStatistic-class,
[39](#)
ScalarIndependenceTestStatistic-class,
[41](#)
SymmetryProblem-class, [51](#)

*Topic **datasets**

alpha, [2](#)
alzheimer, [3](#)
asat, [4](#)
glioma, [12](#)
hohnloser, [13](#)
jobsatisfaction, [21](#)
mercuryfish, [30](#)
neuropathy, [32](#)
ocarcinoma, [34](#)
photocar, [36](#)
rotarod, [40](#)
sphase, [47](#)
treepipit, [58](#)

*Topic **htest**

ContingencyTests, [6](#)
Distribution, [8](#)
IndependenceTest, [15](#)

LocationTests, [21](#)
MarginalHomogeneityTest, [25](#)
MaxstatTest, [27](#)
PermutationDistribution, [35](#)
pvalue-methods, [38](#)
ScaleTests, [42](#)
SpearmanTest, [46](#)
SurvTest, [49](#)
SymmetryTest, [52](#)
SymmetryTests, [53](#)

*Topic **manip**

Transformations, [56](#)

*Topic **methods**

ExactNullDistribution-methods,
[9](#)
expectation-methods, [9](#)
initialize-methods, [20](#)
PermutationDistribution, [35](#)
pvalue-methods, [38](#)
show-methods, [45](#)
statistic-methods, [48](#)

*Topic **misc**

coin, [5](#)

alpha, [2](#)
alzheimer, [3](#)
ansari_test (*ScaleTests*), [42](#)
ansari_trafo (*Transformations*), [56](#)
approximate, [6](#), [16](#), [23](#), [26](#), [28](#), [43](#), [46](#), [52](#),
[54](#)
approximate (*Distribution*), [8](#)
asat, [4](#)
asymptotic, [6](#), [16](#), [23](#), [26](#), [28](#), [43](#), [46](#), [52](#), [54](#)
asymptotic (*Distribution*), [8](#)

chisq_test (*ContingencyTests*), [6](#)

cmh_test (*ContingencyTests*), [6](#)

coerce, IndependenceTestProblem, IndependenceProblem,
(*IndependenceProblem-class*),
[15](#)

- coerce<-, IndependenceTestProblem, IndependenceTest-method
(*IndependenceProblem-class*), 15
- coin, 5
- consal_trafo (*Transformations*), 56
- ContingencyTests, 6
- covariance, 7, 17, 23, 28, 44, 46, 50
- covariance (*expectation-methods*), 9
- covariance, CovarianceMatrix-method
(*expectation-methods*), 9
- covariance, IndependenceLinearStatistic-method
(*expectation-methods*), 9
- covariance, IndependenceTest-method
(*expectation-methods*), 9
- covariance-methods
(*expectation-methods*), 9

- Distribution, 8
- dperm, 7, 17, 23, 28, 44, 46, 50
- dperm (*PermutationDistribution*), 35
- dperm, AsymptNullDistribution-method
(*PermutationDistribution*), 35
- dperm, IndependenceTest-method
(*PermutationDistribution*), 35
- dperm, NullDistribution-method
(*PermutationDistribution*), 35
- dperm-methods
(*PermutationDistribution*), 35

- exact, 16, 22, 43, 52, 54
- exact (*Distribution*), 8
- ExactNullDistribution
(*ExactNullDistribution-methods*), 9
- ExactNullDistribution, ScalarIndependenceTestStatistic-method
(*ExactNullDistribution-methods*), 9
- ExactNullDistribution-methods, 9
- expectation, 7, 17, 23, 28, 44, 46, 50
- expectation
(*expectation-methods*), 9
- expectation, IndependenceLinearStatistic-method
(*expectation-methods*), 9
- expectation, IndependenceTest-method
(*expectation-methods*), 9
- expectation-methods
(*expectation-methods*), 9
- ExpectCovar-class, 11
- ExpectCovarInfluence-class, 11
- exprSet, 16

- f_trafo (*Transformations*), 56
- fligner_test (*ScaleTests*), 42
- fligner_trafo (*Transformations*), 56
- fmaxstat_trafo (*Transformations*), 56
- friedman_test (*SymmetryTests*), 53

- glioma, 12

- hohnloser, 13

- id_trafo (*Transformations*), 56
- independence_test, 7, 9, 58
- independence_test
(*IndependenceTest*), 15
- IndependenceLinearStatistic, 20, 30, 40, 42
- IndependenceLinearStatistic-class, 14
- IndependenceProblem, 15, 19, 20, 30, 40, 42
- IndependenceProblem-class, 15
- IndependenceTest, 15
- IndependenceTest-class, 7, 10, 17, 23, 28, 35, 38, 44, 46, 48, 50
- IndependenceTest-class, 18
- IndependenceTestProblem, 15, 20, 30, 40, 42
- IndependenceTestProblem-class, 19
- IndependenceTestStatistic, 30, 40, 42
- IndependenceTestStatistic-class, 19
- initialize (*initialize-methods*), 20
- initialize, ANY-method
(*initialize-methods*), 20
- initialize, CovarianceMatrix-method
(*initialize-methods*), 20
- initialize, environment-method
(*initialize-methods*), 20

- initialize, IndependenceLinearStatistic-method
 (*initialize-methods*), 20
 initialize, IndependenceProblem-method
 (*IndependenceProblem-class*), 15
 initialize, IndependenceTest-method
 (*IndependenceTest-class*), 18
 initialize, IndependenceTestProblem-method
 (*IndependenceTestProblem-class*), 19
 initialize, IndependenceTestStatistic-method
 (*initialize-methods*), 20
 initialize, MaxTypeIndependenceTestStatistic-method
 (*MaxTypeIndependenceTestStatistic-class*), 29
 initialize, MayTypeIndependenceTest-method
 (*IndependenceTest-class*), 18
 initialize, ModelEnv-method
 (*initialize-methods*), 20
 initialize, QuadTypeIndependenceTest-method
 (*IndependenceTest-class*), 18
 initialize, QuadTypeIndependenceTestStatistic-method
 (*QuadTypeIndependenceTestStatistic-class*), 39
 initialize, ScalarIndependenceTest-method
 (*IndependenceTest-class*), 18
 initialize, ScalarIndependenceTestConfint-method
 (*IndependenceTest-class*), 18
 initialize, ScalarIndependenceTestStatistic-method
 (*ScalarIndependenceTestStatistic-class*), 41
 initialize, signature-method
 (*initialize-methods*), 20
 initialize, SymmetryProblem-method
 (*SymmetryProblem-class*), 51
 initialize, traceable-method
 (*initialize-methods*), 20
 initialize, Variance-method
 (*initialize-methods*), 20
 initialize-methods, 20
 jobsatisfaction, 21
 kruskal_test (*LocationTests*), 21
 chi_test (*ContingencyTests*), 6
 LocationTests, 21
 logrank_trafo (*Transformations*), 56
 MarginalHomogeneityTest, 25
 maxstat_test (*MaxstatTest*), 27
 maxstat_trafo (*Transformations*), 56
 MaxstatTest, 27
 MaxTypeIndependenceTest-class
 (*IndependenceTest-class*), 18
 MaxTypeIndependenceTestStatistic-class, 29
 median_test (*LocationTests*), 21
 median_trafo (*Transformations*), 56
 mercuryfish, 30
 mh_test
 (*MarginalHomogeneityTest*), 25
 neuropathy, 32, 57
 normal_test (*LocationTests*), 21
 normal_trafo (*Transformations*), 56
 NullDistribution-class, 33
 ocarcinoma, 34
 of_trafo (*Transformations*), 56
 oneway_test (*LocationTests*), 21
 PermutationDistribution, 35
 photocar, 36
 pmvnorm, 9
 pperm, 7, 17, 23, 28, 44, 46, 50
 pperm (*PermutationDistribution*), 35
 pperm, AsymptNullDistribution-method
 (*PermutationDistribution*), 35
 pperm, IndependenceTest-method
 (*PermutationDistribution*), 35
 pperm, NullDistribution-method
 (*PermutationDistribution*), 35
 pperm-methods
 (*PermutationDistribution*), 35

- PValue, 33
- pvalue, 7, 17, 23, 28, 44, 46, 50
- pvalue (*pvalue-methods*), 38
- pvalue, IndependenceTest-method (*pvalue-methods*), 38
- pvalue, MaxTypeIndependenceTest-method (*pvalue-methods*), 38
- pvalue, NullDistribution-method (*pvalue-methods*), 38
- pvalue, QuadTypeIndependenceTest-method (*pvalue-methods*), 38
- pvalue, ScalarIndependenceTest-method (*pvalue-methods*), 38
- PValue-class, 37
- pvalue-methods, 38

- qperm, 7, 17, 23, 28, 44, 46, 50
- qperm (*PermutationDistribution*), 35
- qperm, AsymptNullDistribution-method (*PermutationDistribution*), 35
- qperm, IndependenceTest-method (*PermutationDistribution*), 35
- qperm, NullDistribution-method (*PermutationDistribution*), 35
- qperm-methods (*PermutationDistribution*), 35
- QuadTypeIndependenceTest-class (*IndependenceTest-class*), 18
- QuadTypeIndependenceTestStatistic-class, 39

- rotarod, 40

- ScalarIndependenceTest-class (*IndependenceTest-class*), 18
- ScalarIndependenceTestConfint-class (*IndependenceTest-class*), 18
- ScalarIndependenceTestStatistic-class, 41
- ScaleTests, 42
- show, 7, 17, 23, 28, 44, 46, 50
- show, ANY-method (*show-methods*), 45
- show, classRepresentation-method (*show-methods*), 45
- show, genericFunction-method (*show-methods*), 45
- show, IndependenceTest-method (*show-methods*), 45
- show, MaxTypeIndependenceTest-method (*show-methods*), 45
- show, MethodDefinition-method (*show-methods*), 45
- show, MethodWithNext-method (*show-methods*), 45
- show, ModelEnv-method (*show-methods*), 45
- show, ObjectsWithPackage-method (*show-methods*), 45
- show, QuadTypeIndependenceTest-method (*show-methods*), 45
- show, ScalarIndependenceTest-method (*show-methods*), 45
- show, ScalarIndependenceTestConfint-method (*show-methods*), 45
- show, traceable-method (*show-methods*), 45
- show-methods, 45
- spearman_test (*SpearmanTest*), 46
- SpearmanTest, 46
- sphase, 47
- statistic, 7, 17, 23, 28, 44, 46, 50
- statistic (*statistic-methods*), 48
- statistic, IndependenceLinearStatistic-method (*statistic-methods*), 48
- statistic, IndependenceTest-method (*statistic-methods*), 48
- statistic, IndependenceTestStatistic-method (*statistic-methods*), 48
- statistic-methods, 48
- support, 7, 17, 23, 28, 44, 46, 50
- support (*PermutationDistribution*), 35
- support, IndependenceTest-method (*PermutationDistribution*), 35
- support, NullDistribution-method (*PermutationDistribution*), 35

support-methods
 (*PermutationDistribution*),
 35

surv_test, 57

surv_test (*SurvTest*), 49

survdiff, 50

SurvTest, 49

symmetry_test, 54

symmetry_test (*SymmetryTest*), 52

SymmetryProblem-class, 51

SymmetryTest, 52

SymmetryTests, 53

trafo, 16, 17, 52, 53

trafo (*Transformations*), 56

Transformations, 56

treepipit, 58

variance (*expectation-methods*), 9

variance, *CovarianceMatrix*-method
 (*expectation-methods*), 9

variance, *IndependenceLinearStatistic*-method
 (*expectation-methods*), 9

variance, *IndependenceTest*-method
 (*expectation-methods*), 9

variance, *Variance*-method
 (*expectation-methods*), 9

variance-methods
 (*expectation-methods*), 9

wilcox_test (*LocationTests*), 21

wilcoxsign_test (*SymmetryTests*),
 53