Package 'muStat'

February 20, 2015

Type Package

Title Prentice Rank Sum Test and McNemar Test

Version 1.7.0

Date 2010-09-17

Author Knut M. Wittkowski <kmw@rockefeller.edu> and Tingting Song <ttsong@gmail.com>

Maintainer Benedetta Bigio <bbigio@rockefeller.edu>

Depends R (>= 2.11.1), stats

Description Performs Wilcox rank sum test, Kruskal rank sum test, Friedman rank sum test and McNemar test.

License GPL (>= 2)

URL http://mustat.rockefeller.edu

Repository CRAN

Date/Publication 2012-08-23 11:00:20

NeedsCompilation no

R topics documented:

`muStat-package` *Prentice (Friedman/Wilcoxon/Kruskal) Rank Sum Test, Stratified McNemar (TDT, Sign) Test and u/mu-Scores for Multivariate Data*

Description

Performs a generalized Friedman rank sum test with replicated blocked data or, as special cases, a Kruskal-Wallis rank sum test on data following a one-way layout, or a Wilcoxon rank sum test following a one-way layout with only two groups, or a stratified McNemar/sign test using a notion adopted to genetic association studies (TDT). Generates u/mu-scores of univariate and multivariate data to be used with these tests.

Author

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

Maintainer: Tingting Song <ttsong@gmail.com>

Details

Package:	muStat
Type:	Package
Version:	1.6.0
Date:	2010-09-17
URL:	http://mustat.rockefeller.edu
License:	GPL (>= 2)

Copyright

Software Transfer Agreement

(adapted from the NIH Uniform Biological Materials Transfer Agreement, 64 FR 72090)

\

In response to the RECIPIENT's request for the software package **muStat** "SOFTWARE" the PROVIDER asks that the RECIPIENT agree to the following:

1. The SOFTWARE is the property of the PROVIDER and is made available as a service to the research community.
2. THE SOFTWARE WILL NOT BE USED FOR TREATING OR DIAGNOSING HUMAN SUBJECTS.
3. The SOFTWARE will be used for teaching or not-for-profit research purposes only.
4. The SOFTWARE will not be further distributed to others without the PROVIDER's written consent, including any portion of the code incorporated into another work. The RECIPIENT shall refer any request for the SOFTWARE to the PROVIDER. PROVIDER agrees to make the SOFTWARE available, under a separate Agreement to other scientists for teaching or not-for-profit research purposes only.
5. The RECIPIENT agrees to acknowledge the source of the SOFTWARE in any publications reporting use of it.
6. THE PROVIDER MAKES NO REPRESENTATIONS AND EXTENDS NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED. THERE ARE NO EXPRESS OR IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADEMARK, OR OTHER PROPRIETARY RIGHTS. Unless prohibited by law, RECIPIENT assumes all liability for claims for damages against it which may arise from the use of the SOFTWARE.
7. By registering at `http://muStat.rockefeller.edu` ("PROVIDER SCIENTIST WEB SITE") the RECIPIENT obtains the right to receive revisions of the SOFTWARE and related information.
8. By downloading this SOFTWARE, the RECIPIENT agrees to the terms of this AGREEMENT.

Provider Information:

Provider Scientist:	**Knut M. Wittkowski, PhD, DSc** <kmw@rockefeller.edu>
Provider:	**The Rockefeller University**
Address:	**1230 York Ave, New York, NY 10021, U.S.A.**
Name of Authorized Official:	**Kathleen Denis** <denisk@rockefeller.edu>
Title of Authorizing Official:	**Associate Vice President, Technology Transfer**

muS2RC-package *Splus to R Compatibility for package muStat*

Description

muR2SC contains functions that are needed by package muStat and either have different definitions in R and Splus, or defined in Splus while not in R.

Details

Package:	muS2RC
Type:	Package
Version:	1.6.0
Date:	2010-09-17
License:	GPL (>= 2)

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

Maintainer: Tingting Song <ttsong@gmail.com>

Examples

```
x <- c(1, 4, NA, 0, 5)
anyMissing(x)
# [1] TRUE
chkMissing(x)
# [1] TRUE
which.na(x)
# [1] 3
stdev(x, na.rm=TRUE, unbiased=TRUE)
# [1] 2.380476
stdev(x, na.rm=TRUE, unbiased=FALSE)
# [1] 1.904381

c <- 5
ifelse1(c>=0, 1, -1)
# [1] 1
is.inf(Inf)
# [1] TRUE
is.inf(NA)
# [1] FALSE
is.inf(1)
# [1] FALSE

is.number(32)
```

```
# [1] TRUE
is.number(matrix(1:20, nrow=2))
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
# [1,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  TRUE
# [2,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  TRUE
is.number(list(matrix(1:20, nrow=2), 1:4))
# [1] TRUE TRUE
is.number('s')
# [1] TRUE

f1 <- function(x, y) x+y
f2 <- MC(function(x, y) x*y, list(f1=f1))
```

---

muUtil-package *muStat Utility Functions*

Description

This package contains a collection of utility functions for package muStat.

For a complete list, use library(help="muUtil").

Details

Package:	muUtil
Type:	Package
Version:	1.6.0
Date:	2010-09-17
License:	GPL (>= 2)

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

Maintainer: Tingting Song <ttsong@gmail.com>

Examples

```
x <- 5
NoOp(x)
# [1] 5

len(c(1:20))
# [1] 20
```

anyMissing *Check for Missing Values.*

Description

Check if there exist any missing values.

Usage

```
anyMissing(x)
```

Arguments

x	data object

Details

Returns TRUE, if there are any NA's.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

is.na, link{chkMissing}

Examples

```
x <- c(1, 4, NA, 0, 5)
anyMissing(x)
# [1] TRUE
```

bits.per.integer *Internal Size of an integer*

Description

bits.per.integer tells the number of bits that an integer occupies.

Usage

```
bits.per.integer()
```

Details

Assigned to be 32 for the C programs

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
bits.per.integer()

## The function is currently defined as
function() 32
```

---

chkMissing *Check Missing Data*

Description

chkMissing always returns TRUE in R. If in Splus, it returns a logical value, indicating if there exsist NA's in data

Usage

```
chkMissing(...)
```

Arguments

...	object to be tested

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

is.na, link{anyMissing}

Examples

```
x <- c(1, 4, NA, 0, 5)
chkMissing(x)
# [1] TRUE
```

endfunction *End a Function*

Description

endfunction will return an invisible NULL, which could mark the end of a function.

Usage

```
endfunction(text)
```

Arguments

text	any input

Value

Always returns NULL, which will not be printed.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

invisible

Examples

```
endfunction(f)
```

ifelse1 *Conditional Data Selection*

Description

Places values into an object according to the logical values in test.

Usage

```
ifelse1(test, x, y, ...)
```

Arguments

test	logical object. Missing values (NA) are allowed
x	action to be taken if test is TRUE
y	action to be taken if test is FALSE
...	other

Details

NA values in test cause NAs in the result. Compared with ifelse() in Splus, the length of test in ifelse1() is 1, which means ifelse1() will do only one test.

Value

x or y depending on test.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

if, ifelse

Examples

```
c <- 5
ifelse1(c>=0, 1, -1)
# [1] 1
```

`is.inf` *Infinite*

Description

is.inf returns a vector of the same length as the input object, indicating which elements are infinite (not missing).

Usage

```
is.inf(...)
```

Arguments

`...`	object to be tested

Details

is.infinite returns a vector of the same length as x the jth element of which is TRUE if x[j] is infinite (i.e., equal to one of Inf or -Inf). This will be false unless x is numeric or complex. Complex numbers are infinite if either the real and imaginary part is.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

~~objects to See Also as `is.finite`

Examples

```
is.inf(Inf)
# [1] TRUE
is.inf(NA)
# [1] FALSE
is.inf(1)
# [1] FALSE
```

`is.number` *Check Values*

Description

Returns TRUE if the value is finite or infinite, i.e., is neither missing (NA) nor not-a-number (Nan). It replicates the same behavior of the homonymous S+ function.

Usage

```
is.number(...)
```

Arguments

`...`	numeric vector

Details

is.number is TRUE if the value is finite or infinite, i.e., is neither missing (NA) nor not-a-number (NaN).

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
is.number(32)
# [1] TRUE
is.number(matrix(1:20, nrow=2))
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
# [1,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  TRUE
# [2,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  TRUE
is.number('s')
# [1] FALSE
```

`is.orderable` *If a value can be ordered*

Description

is.orderable() returns !is.na()

Usage

```
is.orderable(x)
```

Arguments

x	object to be tested

Details

is.orderable(x) works elementwise when x is a list.

Value

is.orderable returns a logical vector of the same attribute as its argument x

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

is.na

Examples

```
x <- c(1, 4, NA, 0, 5)
is.orderable(x)
# [1]  TRUE  TRUE FALSE  TRUE  TRUE
```

len *Length of an Object*

Description

Get the length of vectors (including lists) and factors, and of any other object for which a method has been defined.

Usage

```
len(...)
```

Arguments

...	object

Details

len() cannot reset the length of a vector, while length() could.

Value

For vectors (including lists) and factors the length is the number of elements. For an environment it is the number of objects in the environment, and NULL has length 0. For expressions and pairlists (including language objects and dotlists) it is the length of the pairlist chain. All other objects (including functions) have length one: note that for functions this differs from S.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
len(c(1:20))
```

MC *Make Closure for functions*

Description

MC makes closures for defining functions in a function.

Usage

```
MC(f, env=NULL)
```

Arguments

f	function
env	a list containing functions to be used in f

Details

MC declares functions to be used in f. When f is defined inside of a function, say fun, it cannot call other functions defined in fun. MC can enclose the functions needed by f and make it possible for f to call other functions defined in fun.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
f1 <- function(x, y) x+y
f2 <- MC(function(x, y) x*y, list(f1=f1))
```

---

mu.AND *Pairwise AND*

Description

mu.AND aggregates vectors of pairwise orderings created by mu.GE in a hierarchical fashion.

Usage

```
mu.AND(GE, frml=NULL)
```

Arguments

GE	matrix each column of which corresponds to one variable anddescribes pairwise orderings
frml	formula describing the hierarchical structure of the variables

Value

mu.AND returns a vector of the same length as the columns of GE, with information about pairwise orderings aggregated according to frml.

Algorithm

```
mu.AND <- function(GE, frml=NULL) {
  <...>
  if (is.null(frml)) {
    <...>
    GE  <- sq.array(GE)
    AND <- GE[,,1]^0
    nNA <- AND[,1]*0
    for (i in 1:dim(GE)[3]) {
      nNA <- nNA + diag(GEi <- GE[,,i])
      AND <- AND * (GEi + (1-GEi)*(1-t(GEi))) }
    return(as.numeric(AND * ((c(nNA)%o%c(nNA))>0))) }

  tkn <- unlist(strsplit(frml,""))
  nok <- attr(regexpr("[0-9,()]+",frml),"match.length")
  <...>
  tmp <- matrix(0, dim(GE)[1]+1, <...>)
  FstFree <- function(tmp) match(TRUE, tmp[1,]==0, nomatch=0)

  level <- i <- 0
  while ((i <- i+1) <= nok) {
    switch( tkn[i],
      "(" = level <- level + 1,
      "," = next,
      ")" = { tmp[1, use <- (tmp[1,]==level)] <- 0
              tmp[, FstFree(tmp)] <- c(level <- level-1, mu.AND(tmp[-1, use]))},
      { num <- as.numeric(substring(
                frml,i,i<-i-1+regexpr("[,)]",substring(frml,i+1)))
        <...>
        tmp[,FstFree(tmp)] <- c(level, GE[, num]) } ) }
  return(tmp[-1,1])
}
```

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

Examples

```
mu.AND(mu.GE(matrix(1:60, , 3)))
a.1 <- 1:10
a.2 <- 3:12
b <- c(1:5,2:6)
mu.AND(mu.GE(cbind(a.1,a.2,b)), frml="((1,2),3)")
```

mu.dbinom *Binomial Distribution*

Description

Probability mass function (density) of the binomial distribution.

Usage

```
mu.dbinom(x, size, prob, log = FALSE)
```

Arguments

x	vector of quantiles. Missing values (NAs) are allowed.
size	vector of (positive integer) numbers of coin flips for which the Binomial distribution measures the number of heads.
prob	vector of probabilities of a head. If length(x) is larger than 1, then length(x) random values are returned.
log	logical; if TRUE, probabilities p are given as log(p).

Details

```
mu.dbinom <- function(x, size, prob, log=FALSE)
  if (size==0) 1 else dbinom(x, size, prob, log=FALSE)
```

Value

Returns the density of bin(size, prob) at x.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

See Also

dbinom

Examples

```
dbinom(100, 10, 3/10, log=FALSE)
```

mu.GE *GE Matrix*

Description

mu.GE returns a matrix with each element denoting the logical value if one value is greater than or equal to another.

Usage

```
mu.GE(x, y=x)
```

Arguments

x	data matrix, variables as columns
y	second data matrix, optional

Details

The (i,j) entry of GE matrix is 1 if $x_i \geq y_j$, 0 otherwise. The square matrix GE is stored by column in a vector.

Value

a vector which contains the GE matrix.

Algorithm

```
mu.GE <- function(x, y=x) {
  <...>
  if (length(y)>1)
    apply(rbind(x,y),2,mu.GE,nrow(x))
  else
    as.numeric(NAtoZer(outer(x[1:y],x[-(1:y)],">=")))
}
```

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

Examples

```
a <- c(4, 2, 5, 1, 4, NA, 6)
mu.GE(a)
```

mu.rank *Ranks of Data*

Description

Returns a vector of the (mid-) ranks of the input.

Usage

```
mu.rank(x, na.last = TRUE, na.rm=Inf)
mu.rank.nna(x)
```

Arguments

x	numeric vector. Missing values (NA) are allowed for mu.rank but not for mu.rank.nna
na.last	vector with one element. If TRUE, NAs are put last, if FALSE, NAs are put first, if NA, NAs are handled according to na.rm; "keep" is equivalent to NA and na.rm = F.
na.rm	logical flag, indicating if missing values (NA) should be removed (TRUE) or not (FALSE) in the output. If NA, NAs in x are not allowed and na.last is ignored. The default for na.rm is TRUE if na.last = NA and FALSE else.

Details

mu.rank is faster than rank. The treatment of missing values is controlled by both na.last and na.rm.

Value

the ranks; i.e., the i-th value is the rank of x[i]. In case of ties, average ranks is returned.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

See Also

rank

Examples

```
a <- c(4, 2, 5, 1, 4, NA, 6)
mu.rank(a) # default: na.last=TRUE, na.rm=FALSE
# [1] 3.5 2.0 5.0 1.0 3.5 7.0 6.0
mu.rank(a,na.last=NA) # default: na.rm=TRUE
# [1] 3.5 2.0 5.0 1.0 3.5     6.0
mu.rank(a,na.last=NA,na.rm=FALSE)
#     3.5 2.0 5.0 1.0 3.5  NA 6.0
```

```
# Spearman's rank correlation between two sets of testscores
a <- c(4, 2, 5, 1, 4, NA, 6)
b <- c(4, 2, 5, NA, 4, 5, 6)

cor(a, b, if(is.R()) "complete.obs" else "available")
# [1] 0.8241688
cor(a, b, if(is.R()) "pairwise.complete.obs" else "omit")
# [1] 1

cor(rank(a), rank(b))
# [1] 0.1651446
cor(mu.rank(a, na.last=NA, na.rm=FALSE),
    mu.rank(b, na.last=NA, na.rm=FALSE),
    if(is.R()) "complete.obs" else "available")
# [1] 0.8523852
cor(mu.rank(a, na.last=NA, na.rm=FALSE),
    mu.rank(b, na.last=NA, na.rm=FALSE),
    if(is.R()) "pairwise.complete.obs" else "omit")
# [1] 0.9953452
cor(rank(a[!is.na(a*b)]), rank(b[!is.na(a*b)]))
# [1] 1
```

`mu.Sums` *u-Scores and Weights*

Description

mu.Sums computes u-scores and weights from a partial ordering.

Usage

```
mu.Sums(GE, dsgn=1, wght=TRUE)
mu.wScr  (x, y=NULL, frml=NULL, dsgn=1, wght=TRUE)
mu.score (x, y=NULL, frml=NULL, dsgn=1, wght=FALSE)
mu.weight(x, y=NULL, frml=NULL, dsgn=1, wght=TRUE)
```

Arguments

`GE`	partial ordering
`x`	data matrix, variables as columns
`y`	data matrix, variables as columns
`frml`	see `mu.AND`,
`dsgn`	design of the experiment.
`wght`	logical flag, if weights should be calculated.

Value

score	u-scores
weight	weights
nBelow	number of observations that are inferior
nAbove	number of observations that are superior
nEqual	number of observations that are equivalent

Algorithm

```
mu.Sums <- function(GE, dsgn=1, wght=TRUE) {
    ICW <- function(GE,dsgn) {
        wgt <- colSums(GE|t(GE))        # 0: orig NA, 1: like NA, >1: no NA
        sqrt(wgt*(wgt>1)/if (dsgn!=1) colSums(dsgn) else nrow(GE))
    }
    GE <- sq.matrix(GE)
    nE <- colSums(GE*t(GE))
    nA <- colSums(GE) - nE
    nB <- rowSums(GE) - nE
weight <- ifelse1(wght, ICW(GE,dsgn), rep(1, length(nE)))
list(score  = (nB-nA) * ifelse(weight==0,NA,1),
weight = weight,
nBelow = nB,
nAbove = nA,
nEqual = nE)
}

mu.wScr <- function(x, y=NULL, frml=NULL, dsgn=1, wght=TRUE)
mu.Sums(mu.AND(mu.GE(x, y), frml), dsgn=dsgn, wght=wght)

mu.score  <- function(x, y=NULL, frml=NULL, dsgn=1, wght=FALSE)
mu.Sums(mu.AND(mu.GE(x, y), frml), dsgn=dsgn, wght=wght)$score

mu.weight <- function(x, y=NULL, frml=NULL, dsgn=1, wght=TRUE)
mu.Sums(mu.AND(mu.GE(x, y), frml), dsgn=dsgn, wght=wght)$weight
```

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>, Tingting Song <ttsong@gmail.com>

Examples

```
mu.Sums(mu.GE(1:100))
a.1 <- 1:10
a.2 <- 3:12
b <- c(1:5,2:6)
mu.wScr(cbind(a.1,a.2,b), frml="((1,2),3)")
```

NAtoZer *Dealing With Missing Values*

Description

Any missing values in input data will be converted to 0.

Usage

```
NAtoZer(x)
```

Arguments

x	data object

Value

Returns the input data object x with all missing values replaced by 0.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
NAtoZer(c(2, 4, NA, 5))
# [1] 2 4 0 5
```

NoOp *No Operation*

Description

NoOp does nothing to the input.

Usage

```
NoOp(x)
```

Arguments

x	input

Value

x is returned.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
x <- 5
NoOp(x)
# [1] 5
```

NoWarn *Suppress Warnings*

Description

NoWarn executes x and temporarily disables the display of warnings.

Usage

```
NoWarn(x)
```

Arguments

x	input

Value

invisible(NULL)

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
NoWarn(0)
```

prentice.test *Prentice (Friedman/Wilcoxon/Kruskal) Rank Sum Test*

Description

Performs a generalized Friedman rank sum test with replicated blocked data or, as special cases, a Kruskal-Wallis rank sum test on data following a one-way layout or a Wilcoxon rank sum test following a one-way layout with only two groups.

Usage

```
prentice.test(y, groups, blocks = NULL,
  score = "rank", blkwght = "prentice", condvar = TRUE,
  alternative = "two.sided", mu = 0, paired = FALSE,
  exact = NULL, correct = FALSE, df = -1, warn = 0, optim = TRUE)
mu.wilcox.test(y, groups, blocks = NULL, score = "rank",
  paired = FALSE, exact = TRUE, correct = TRUE, ...)
mu.kruskal.test(y, groups, blocks, ... )
mu.friedman.test(y, groups, blocks, ... )
```

Arguments

y	a numeric vector of data values, NAs are used to compute block weights with incomplete block data (see blkwght), but will be removed with one-way designs. Blocks with observations in only one group will be removed. Infs are allowed, and are not removed as they are rankable. Required.
groups	factor or category object of the same length as y, giving the group (treatment) for each corresponding element of y. NAs are allowed and observations with NA in groups will be used for scoring other observations, treating them as if they were observations in an additional (fictional) group. If not a factor or category object, it will be coerced to one. Required.
blocks	factor or category object of the same length as y, giving the block membership for each corresponding element of y. Observations with NA in blocks will be removed. If not a factor or category object, it will be coerced to one.
score	character or function object, giving the score function to be used. If NULL, y is assumed to already have been scored, e.g., by marginal likelihood scores (Wittkowski 1992) or u-scores (Wittkowski 2004) for multivariate data. If a function, it is applied to the ranks of y (Lehmann, 1951), if character, • "rank" or "wilcoxon" are equivalent to the default, • "normal" or "vanderwaerden" are ... • ... (to be continued)
blkwght	character object indicating the weights to apply to blocks depending on the ratio between planned size (including NAs) and observed size (excluding NAs). Options are • "prentice",

- "klotz",
- "skillingsmack",
- "rai"

see Wittkowski (1988) and Alvo-Cabilio (2005) for details.

condvar	if FALSE, the variance is conditional on ("corrected for") the observed ties, otherwise, the variance is the expected variance in the absence of ties, see Wittkowski (1988, 1998) and Randles (2001) for details.
df	if -1, the degrees of freedom are computed from the observed data, if 0, the degrees of freedom are computed from the planned number of groups, if >0, the parameter value is taken as the degrees of freedom.
alternative	character string, one of"greater", "less" or "two.sided", indicating the specification of the alternative hypothesis. For Wilcoxon test only.
mu	a single number representing the value of the mean or difference in means specified by the null hypothesis.
paired	if TRUE, the Wilcoxon signed rank test is computed. The default is the Wilcoxon rank sum test.
exact	if TRUE the exact distribution for the test statistic is used to compute the p-value if possible.
correct	if TRUE a continuity correction is applied to the normal approximation for the p-value.
warn	no warnings will be given if warn is -1
optim	if FALSE, the generic algorithm (see Algorithm) is always used, for testing and teaching purposes only if TRUE, faster algorithms are used when available.
...	further arguments to be passed to or from methods.

Details

prentice.test is approximately twice as fast as friedman.test or kruskal.test. In some cases, the Kruskal-Wallis test reduces to the Wilcoxon Rank-sum test. Thus, prentice.test allows the additional parameters mu, paired, exact, and correct, to be entered, and passed. To ensure consistency of the results between wilcox.test and kruskal.test, the default for correct is FALSE in either case.

Value

A list with class "htest" containing the following components:

statistic	the value of chi-squared statistic, with names attribute "statistic: chi-square". See section DETAILS for a definition.
parameter	the degrees of freedom of the asymptotic chi-squared distribution associated with statistic. Component parameters has names attribute "df".
p.value	the asymptotic or exact p-value of the test.
method	a character string giving the name of the method used.
data.name	a character string (vector of length 1) containing the actual names of the input arguments y, groups, and blocks.

Null Hypothesis

The null hypothesis is that for any two observations chosen randomly from the same block, the probability that the first is larger than the second is the same as the probability that it is smaller.

Test Assumptions

The errors are assumed to be independent and identically distributed. The returned p.value should be interpreted carefully. It is only a large-sample approximation whose validity increases with the size of the smallest of the groups and/or the number of blocks.

Algorithm

```
prentice.test <- function(
  y,
  groups,
  blocks  = NULL,
  score   = "rank",       # NULL: y already scored
  blkwght = "prentice",   # block weights
  <...>)
{
  <...>

  m  <- xTable(blocks,groups)
  <...>
  p  <- dim(m)[2]-1 # planned number of groups

  y.ok   <- <...>

  y      <- y     [y.ok]
  groups <- groups[y.ok]
  blocks <- blocks[y.ok]
  M  <- xTable(blocks,groups)

  <...>

  mi <- rowSums(m)
  Mi <- rowSums(M)
  Wi <- switch(tolower(blkwght),
    prentice      = (mi+1),
    klotz         = (Mi+1),
    skillingsmack = sqrt(Mi+1),
    rai           = (Mi+1)/Mi,
    <...>)
  Bijk <- Wi[blocks]

  Tijk <- Centered(
    Score(FUNByIdx(y,blocks,wRank,na.any=FALSE)/(Mi[blocks]+1)),
    blocks, Mi) * Bijk
```

```
  T1 <- qapply(Tijk,groups,sum)

  A0i2 <- (1/(Mi-1))*qapply(Tijk^2,blocks,sum)
  V0   <- structure(dim=c(P,P), A0i2 %*% (
            t(apply(M,1,   function(x,P) diag(x))) - (1/Mi) *
            t(apply(M,1,MC(function(x) outer1(x),list(outer1=outer1))))))
  V1   <- ginv(V0)

  W    <- as.numeric(T1 %*% V1 %*% T1)
  df.W <- attr(V1,"rank")
  p.W  <- 1 - pchisq(W, df.W)
}
```

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

References

Friedman, M. (1937) *Journal of the American Statistical Association*, **32**: 675-701.

Lehmann, E. L. (1951) *Annals of Mathematical Statistics*, **22**: 165-179.

Kruskal, W. H. and Wallis, W. A. (1952) *Journal of the American Statistical Association*, **47**: 583-631.

Hajek, J. and Sidak, Z. (1967) *Theory of rank tests*, New York, NY: Academic.

Hollander, M. and Wolfe, D. A. (1973). *Nonparametric Statistical Methods*. New York, NY: John Wiley.

Lehmann, E. L. (1975). *Nonparametrics: Statistical Methods Based on Ranks*. Oakland, CA: Holden-Day.

Prentice, M. J. (1979) *Biometrika*, **66**: 167-170.

Wittkowski, K. M. (1988) *Journal of the American Statistical Association*, **83**: 1163-1170.

Alvo, M. and Cabilio, P. (2005) *Canadian Journal of Statistics-Revue Canadienne De Statistique*, **33**: 115-129.

Wittkowski, K. M. (1992) *Journal of the American Statistical Association*, **87**: 258.

Wittkowski, K. M. (1998) *Biometrics*, **54**: 789¡§C791.

Randles, H. R. (2001) *The American Statistician*, **55**: 96-101.

Wittkowski, K. M., Lee, E., Nussbaum, R., Chamian, F. N. and Krueger, J. G. (2004) *Statistics in Medicine*, **23**: 1579-1592.

See Also

wilcox.test, kruskal.test, friedman.test, rank, aov

Examples

```
# friedman.test examples

  treatments <- factor(rep(c("Trt1", "Trt2", "Trt3"), each=4))
  people <- factor(rep(c("Subj1", "Subj2", "Subj3", "Subj4"), 3))
  y <- c(0.73,0.76,0.46,0.85,0.48,0.78,0.87,0.22,0.51,0.03,0.39,0.44)
  print(   friedman.test(y, treatments, people))
  print(mu.friedman.test(y, treatments, people))

  # Now suppose the data is in the form of a matrix,
  #   rows are people and columns are treatments.
  # Generate 'ymat' and the factor objects:

  ymat <- matrix(c(0.73,0.76,0.46,0.85,0.48,0.78,0.87,0.22,0.51,
        0.03,0.39,0.44), ncol=3)
  bl <- factor(as.vector(row(ymat)))
  gr <- factor(as.vector(col(ymat)))
  print(   friedman.test(ymat, gr, bl))  # same answer as above
  print(mu.friedman.test(ymat, gr, bl))

# kruskal.test examples

  # Data from Hollander and Wolfe (1973), p. 116
  holl.y <- c(2.9,3.0,2.5,2.6,3.2,3.8,2.7,4.0,2.4,2.8,3.4,3.7,2.2,2.0)
  holl.grps <- factor(c(1,1,1,1,1,2,2,2,2,3,3,3,3,3),
      labels=c("Normal Subjects","Obstr. Airway Disease","Asbestosis"))
  print(   kruskal.test(holl.y, holl.grps))
  print(mu.kruskal.test(holl.y, holl.grps))

  # Now suppose the data is in the form of a table already,
  # with groups in columns; note this implies that group
  # sizes are the same.

  tab.data <- matrix(c(.38,.58,.15,.72,.09,.66,.52,.02,.59,.94,
        .24,.94,.08,.97,.47,.92,.59,.77), ncol=3)
  tab.data

  y2 <- as.vector(tab.data)
  gr <- factor(as.vector(col(tab.data)))   # Groups are columns
  print(   kruskal.test(y2, gr))
  print(mu.kruskal.test(y2, gr))

# wilcox.test examples

  x <- c(8.2, 9.4, 9.6, 9.7, 10.0, 14.5, 15.2, 16.1, 17.6, 21.5)
  y <- c(4.2, 5.2, 5.8, 6.4, 7.0, 7.3, 10.1, 11.2, 11.3, 11.5)
  print(   wilcox.test(x,y))
  print(mu.wilcox.test(x,y))
  print(   wilcox.test(x,y, exact=FALSE))
  print(mu.wilcox.test(x,y, exact=FALSE))
  print(   wilcox.test(x,y, exact=FALSE, correct=FALSE))
```

```
  print(mu.wilcox.test(x,y, exact=FALSE, correct=FALSE))
  xy <- c(x,y)
  groups <- c(rep(1,length(x)),rep(2,length(y)))
  print(prentice.test(xy,groups,exact=FALSE, correct=FALSE))

# compare speed

  if (is.R()) sys.time <- function (...) system.time(...)

  n <- 1000
  data <- runif(30*n)
  grps <- c(rep(1,10*n),rep(2,8*n),rep(3,12*n))

  print(sys.time(    kruskal.test( data,grps)              ))
  print(sys.time( mu.kruskal.test( data,grps,optim=FALSE)  ))
  print(sys.time(    prentice.test(data,grps)              ))

  data <- runif(600)
  grps <- rep(1:6,each=100)
  blks <- rep(1:100,length.out=length(data))

  print(sys.time(    friedman.test(data,grps,blks)             ))
  print(sys.time( mu.friedman.test(data,grps,blks,optim=FALSE) ))
  print(sys.time(    prentice.test(data,grps,blks)             ))

  data <- runif(50000)
  grps <- rep(1:2,each=25000)
  Wx <- data[grps==1]
  Wy <- data[grps==2]

  print(sys.time(    wilcox.test(Wx,Wy)                    ))
  print(sys.time( mu.wilcox.test(Wx,Wy,optim=FALSE)        ))
  print(sys.time(    prentice.test(data,grps)              ))
```

RCRng *Data Extract*

Description

Returns first and last emelments of a data object.

Usage

```
RCRng(n)
```

Arguments

n	data object

Value

An vector which contains first and last values of x.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
matrixB <- matrix(100:1,ncol=100,nrow=100)
RCRng(matrixB)
# [1] 100   1
```

SMN.pvalue *Stratified McNemar (TDT, sign) test for association studies*

Description

Performs an asymptotic or exact stratified McNemar/sign test using a notation adopted to genetic association studies.

Usage

```
SMN.pvalue(pP,qP, pX=0,qX=0, pQ=0,qQ=0, wP=0.25,wQ=0.25, exact=NULL)
TDT.pvalue(pP,qP, pX,xX,qX, pQ,qQ,               exact=FALSE)
DMM.pvalue(pP,qP,    xX,           condvar=TRUE, exact=FALSE)
MCN.pvalue(pP,qP,                                exact=FALSE)
```

Arguments

pP	a numeric value, number of children homozygous for allele "P" ("PP") among parents of mating types "PP" and "PQ". Corresponds to "a" or "0/1" in a classical 2×2 table. Required.
qP	a numeric value, number of heterozygous children ("PQ") among parents of mating types "PP" and "PQ". Corresponds to "c" or "1/0" in a classical 2×2 table. Required.
pX	a numeric value, number of children homozygous for allele "P" ("PP") among parentso of mating types "PQ" and "PQ" (both heterozygous). Corresponds to "a" or "0/1" in a classical 2×2 table.
xX	a numeric value, number of heterozygous children ("PQ") among parentso of mating types "PQ" and "PQ" (both heterozygous). Corresponds to the sum of "b" and "d" or "0/0" and "1/1" in a classical 2×2 table.
qX	a numeric value, number of children homogzygous for allele "Q" ("QQ") among parentso of mating types "PQ" and "PQ" (both heterozygous). Corresponds to "c" or "1/0" in a classical 2×2 table.

qQ	a numeric value, number of heterozygous children ("PQ") among parents of mating types "PQ" and "QQ". Corresponds to "c" or "1/0" in a classical 2×2 table.
pQ	a numeric value, number of children homozygous for allele "Q" ("QQ") among parents of mating types "PQ" and "QQ". Corresponds to "c" or "0/1" in a classical 2×2 table.
wP	relative weight to be assigned to children with one "PP" parent. Default: `.25`.
wQ	relative weight to be assigned to children with one "QQ" parent. Default: `.25`.
condvar	if `FALSE`, the variance is conditional on ("corrected for") the observed ties, otherwise, the variance is the expected variance in the absence of ties, see Wittkowski (1988, 1998) and Randles (2001) for details.
exact	if `TRUE`, the exact p-value is computed unless the product of block sizes exceets `10^6`, if `NULL`, the exact p-value is computed if the sum of block sizes is below `100`, if `FALSE`, the exact p-value is never computed.

Details

`TDT.pvalue` is given for historical purposes only. It should be deprecated, because it is based on the unrealistic assumption that the individual *alleles* (rather than *children*) are randomly chosen from the population of transmitted alleles.

Choosing weights `(wP, wQ)` as `(.00, .50)` or `(.50, .00)` maximises the sensitivity of the test to detect dominant or recessive alleles, respectivel.

Value

p.value	the asymptotic or exact p-value of the test.

Null Hypothesis

The null hypothesis is that for any two observations chosen randomly from the same block (parental mating type), the probability that it falls into the first category ("PP", "PP", or "PQ", respectively) is the same as the probability that it falls into the second category ("PQ", "QQ", or "QQ", respectively).

Test Assumptions

Except for `TDT.pvalue`, the children are assumed to be randomly chosen from the population of children born to parents with the same parental mating type. The asymptotic p.value should be interpreted carefully. It is only a large-sample approximation whose validity increases with the size of the smallest of the groups and/or the number of blocks.

Algorithm

```
SMN.pvalue <- function(pP,qP, pX=0,qX=0, pQ=0,qQ=0,
      wP=.25,wQ=.25, exact=NULL) {
  M <- function(P,X,Q, wP,wQ, e = 1, op = "+", f = function(x) x) {
    O3 <- function(X,Y,Z,Op) matrix(outer(outer(X,Y,Op),Z,Op))
    O3(wP^e*f(P), (1-wP-wQ)^e*f(X), wQ^e*f(Q), op) }
```

```
  exact <- (dP<-pP-qP)+(dX<-pX-qX)+(dQ<-pQ-qQ)<100
            && is.null(exact) || exact
  if (((nP<-pP+qP)*(nX<-pX+qX)*(nQ<-pQ+qQ)>10^6) || !exact )
    return( 1-pchisq(
      M(dP,dX,dQ, wP,wQ)^2/              # Eq. (1) in Wittkowski (2002)
      M(nP,nX,nQ, wP,wQ, 2),1)[1])       # Eq. (2) in Wittkowski (2002)
  else {
    tb <- cbind(
      M(nP,nX,nQ, wP,wP, 0,"*", function(n) mu.dbinom(0:n, n, .5)),
      M(nP,nX,nQ, wP,wQ, 1,"+", function(n) (0:n)-(n:0) )^2)
    return(1-sum(tb[tb[,2]<c(M(dP,dX,dQ, wP,wQ)^2),1])) }
}
```

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

References

Dixon, W.J., Mood, A.M. (1946) *J Am Statist Assoc* **41**: 557-566

McNemar, Q (1947) *Psychometrica* **12**: 153-157

Dixon, W.J., Massey, F.J.J. (1951) *An Introduction to Statistical Analysis*. New York, NY: McGraw-Hill

Wittkowski, K. M. (1988) *Journal of the American Statistical Association*, **83**: 1163-1170.

Wittkowski, K. M. (1998) *Biometrics*, **54**: 789-C791

Spielman, R.S., McGinnis, R.E., Ewens, W.J. (1993) *Am J Hum Genet* **52**: 506-516.

Wittkowski, K.M., Liu, X. (2002) Hum Hered **54**: 157-164, **58**: 59-62

See Also

friedman.test, binom.test, chisq.test, mcnemar.test, mantelhaen.test

Examples

```
SMN.pvalues <- function(n, wP = 0.25, wQ = 0.25) {
  print(SMN.pvalue(
    n[1,1],n[1,2], n[2,1], n[2,3], n[3,2],n[3,3], wP, wQ, exact = FALSE))
  print(SMN.pvalue(
    n[1,1],n[1,2], n[2,1], n[2,3], n[3,2],n[3,3], wP, wQ, exact = TRUE))
}
TDT.pvalues <- function(n){
  print(TDT.pvalue(
    n[1,1],n[1,2], n[2,1],n[2,2],n[2,3], n[3,2],n[3,3], exact = FALSE))
  print(TDT.pvalue(
    n[1,1],n[1,2], n[2,1],n[2,2],n[2,3], n[3,2],n[3,3], exact = TRUE))
}
```

```
wP <- 0.25;  wQ <- 0.25

n <- matrix(c(
    1,3,0,   wP,
    1,0,1,(1-wP-wQ),
    0,3,1,      wQ), ncol=4, byrow=TRUE)

SMN.pvalues(n)
TDT.pvalues(n)

n[3,2] <- 1
n[3,3] <- 3;  SMN.pvalues(n);  TDT.pvalues(n)

n[2,2] <- 1;  SMN.pvalues(n);  TDT.pvalues(n)

n[2,2] <- 3;  TDT.pvalues(n)
n[2,2] <- 2;  TDT.pvalues(n)
n[2,3] <- 3;  TDT.pvalues(n)

SMN.pvalues(n, .25, .25)
SMN.pvalues(n, .00, .50)
SMN.pvalues(n, .50, .00)
```

`sq.array` *Produce Square Array or Square Matrix*

Description

Convert input data to a square array or a square matrix.

Usage

```
sq.array(x)
sq.matrix(x)
```

Arguments

`x`	data object. If `x` is a vector, length of `x` should be a square number. Otherwise, the second dimension of `x` should be a square number.

Value

A square matrix with elements coming from `x`.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
sq.matrix(1:25)
sq.array(matrix(1:50, ,2))
```

---

stdev *Standard Deviation*

Description

stdev computes the standard deviation of the values in x. If na.rm is TRUE then missing values are removed before computation proceeds. If x is a matrix or a data frame, a vector of the standard deviation of the columns is returned. If unbiased is TRUE then the sample standard deviation is returned, else the population standard deviation is returned.

Usage

```
stdev(x, na.rm, unbiased)
```

Arguments

x	a numeric vector, matrix or data frame
na.rm	logical value indicating if missing values should be removed.
unbiased	whether to return biased or unbiased standard deviation

Value

Standard deviation of x.

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

sd, var

Examples

```
x <- c(1, 4, NA, 0, 5)
stdev(x, na.rm=TRUE, unbiased=TRUE)
# [1] 2.380476
stdev(x, na.rm=TRUE, unbiased=FALSE)
# [1] 1.904381
```

`ULPrint` *Print Data*

Description

Prints a data object according to the specified size.

Usage

```
ULPrint(aData, nRows=10, nCols=10, Title="")
```

Arguments

`aData`	data
`nRows`	no. of rows to print
`nCols`	no. of cols to print
`Title`	title of output

Value

aData is printed according to nRows and nCols

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

Examples

```
matrixA <- matrix(1:100, nrow=100, ncol=100)
ULPrint(matrixA)
#   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
# [1,]    1    1    1    1    1    1    1    1    1     1
# [2,]    2    2    2    2    2    2    2    2    2     2
# [3,]    3    3    3    3    3    3    3    3    3     3
# [4,]    4    4    4    4    4    4    4    4    4     4
# [5,]    5    5    5    5    5    5    5    5    5     5
# [6,]    6    6    6    6    6    6    6    6    6     6
# [7,]    7    7    7    7    7    7    7    7    7     7
# [8,]    8    8    8    8    8    8    8    8    8     8
# [9,]    9    9    9    9    9    9    9    9    9     9
# [10,]   10   10   10   10   10   10   10   10   10    10
# ...
```

which.na *Determine Which Values are Missing Values*

Description

which.na returns an integer vector describing which values in the input vector, if any, are missing

Usage

```
which.na(x)
```

Arguments

x	a data object

Value

integer vector containing indices of elements in x which are missing. If there are no missing values, the functions return an integer vector of length 0 (numeric(0)).

Author(s)

Knut M. Wittkowski <kmw@rockefeller.edu>

See Also

is.na

Examples

```
x <- c(1, 4, NA, 0, 5)
which.na(x)
# [1] 3
```

Index