Package 'asht'

August 24, 2023

Type Package

Title Applied Statistical Hypothesis Tests

Version 1.0.1

Date 2023-08-24

Author Michael P. Fay

Maintainer Michael P. Fay <mfay@niaid.nih.gov>

Description Gives some hypothesis test functions (sign test, median and other quantile tests, Wilcoxon signed rank test, coefficient of variation test, test of normal variance, test on weighted sums of Poisson [see Fay and Kim <doi:10.1002/bimj.201600111>], sample size for t-tests with different variances and non-equal n per arm, Behrens-Fisher test, nonparametric ABC intervals, Wilcoxon-Mann-Whitney test [with effect estimates and confidence intervals, see Fay and Malinovsky <doi:10.1002/sim.7890>], two-sample melding tests [see Fay, Proschan, and Brittain <doi:10.1111/biom.12231>], one-way ANOVA allowing var.equal=FALSE [see Brown and Forsythe, 1974, Biometrics]), prevalence confidence intervals that adjust for sensitivity and specificity [see Lang and Reiczigel, 2014 <doi:10.1016/j.prevetmed.2013.09.015>] or Bayer, Fay, and Graubard, 2023 <doi:10.48550/arXiv.2205.1349
cus is on hypothesis tests that have compatible confidence intervals, but some functions only have confidence intervals (e.g., prevSeSp).

License GPL-3

Depends stats, exact2x2 (>= 1.6.4), exactci, bpcp, coin

Imports perm, ssanv

Suggests bootstrap

NeedsCompilation no

RoxygenNote 7.2.3

Repository CRAN

Date/Publication 2023-08-24 14:50:02 UTC

R topics documented:

asht-package 2
abcnonHtest 3

ama1c1cpg 4
anovaOneWay 5
bfTest 7
cvTest 9
meldCD 10
meldtTest 13
metaNorm 15
prevSeSp 17
quantileTest 20
signTest 21
simulateSS 23
tukeyWelsch 25
var1Test 28
wmwTest 29
WprevSeSp 33
wspoissonTest 35
wsrTest 38

Index **41**

`asht-package` *Applied Statistical Hypothesis Tests*

Description

Test and confidence intervals for some applied statistical hypothesis tests.

Details

A collection of statistical hypothesis tests, with a focus on non-asymptotic tests. Some tests are `medianTest` for exact tests and confidence intervals about a median, `quantileTest` which generalizes `medianTest` for other quantiles besides the median, `signTest` to run the exact sign test, `bfTest` to run the Behrens-Fisher test, `abcnonHtest` to calculate ABC intervals and tests, `wmwTest` to run the Wilcoxon-Mann-Whitney test (i.e., Wilcoxon rank sum test, or Mann-Whitney U test) and calculate confidence intervals on the Mann-Whitney parameter. In rare cases, the function only gives a confidence interval and and estimate and does not test a specific hypothesis (see `prevSeSp` which estimates prevalence accounting for sensitivity and specificity).

Author(s)

Michael P. Fay

Maintainer: Michael P. Fay <mfay@niaid.nih.gov>

abcnonHtest *Nonparametric ABC (Approximate Bootstrap Confidence) intervals.*

Description

A hypothesis testing function using the nonparametric ABC intervals.

Usage

```
abcnonHtest(x, tt, nullValue = NULL, conf.level = 0.95,
  alternative = c("two.sided", "less", "greater"), epsilon = 0.001, minp = 0.001)
```

Arguments

x	the data. Must be either a vector, or a matrix whose rows are the observations
tt	function defining the parameter in the resampling form tt(p,x), where p is the vector of proportions and x is the data
nullValue	null value of the parameter for the two-sided hypothesis test, or boundary of null parameter space for one-sided ones
conf.level	confidence level for interval
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
epsilon	optional argument specifying step size for finite difference calculations
minp	minimum p-value (used in uniroot search to give a bound, toe two.sided alternatives actual minimum is 2*minp)

Details

Calculates the nonparametric ABC confidence interval of DiCiccio and Efron (1992). See also Efron and Tibshirani (1993).

The p-values are calculated by solving for confidence limit that just touches the nullValue. If it is outside of the range (minp, 1-minp) for one-sided p-values, then it is set to minp. If it is outside the range (2*minp, 1- 2*minp) for two-sided p-values, then it is set to 2*minp.

Value

A value of class "htest" containing the following components:

p.value	p-value for test defined by alternative and nullValue
estimate	estimate of the parameter, calculated using x and the tt function
conf.int	confidence interval for the parameter associated with tt
null.value	the nullValue (or null boundary) for the hypothesis test
alternative	a character string describing the alternative hypothesis
method	a character string describing the kind of test
data.name	a character string giving the name of the data and the function

Author(s)

the function is modification of abcnon in the bootstrap R package, originally written by Rob Tibshirani, modifications by M.P. Fay

References

DiCiccio, T and Efron, B (1992). More accurate confidence intervals in exponential families. Biometrika 79: 231-245.

Efron, B and Tibshirani, RJ (1993). An introduction to the bootstrap. Chapman and Hall: New York.

See Also

See also abcnon.

Examples

```
# compute abc intervals for the mean
x <- c(2,4,12,4,6,3,5,7,6)
theta <- function(p,x) {sum(p*x)/sum(p)}
## smallest p-value is 2*minp for two-sided alternatives
abcnonHtest(x, theta, nullValue=0)
## test null at 95% confidence limit is like just barely
## rejecting at the two-sided 5% level, so p-value is 0.05
abcnonHtest(x, theta, nullValue=4.072772)
# compute abc intervals for the correlation
set.seed(1)
x <- matrix(rnorm(20),ncol=2)
theta <- function(p, x)
{
    x1m <- sum(p * x[, 1])/sum(p)
    x2m <- sum(p * x[, 2])/sum(p)
    num <- sum(p * (x[, 1] - x1m) * (x[, 2] - x2m))
    den <- sqrt(sum(p * (x[, 1] - x1m)^2) *
              sum(p * (x[, 2] - x2m)^2))
    return(num/den)
}
abcnonHtest(x, theta)
## compare with
## Not run:
library(bootstrap)
abcnon(x, theta, alpha=c(.025,.975))$limits[,"abc"]
## End(Not run)
```

---

ama1c1cpg *Three arm phase 1 malaria vaccine trial*

Description

Growth inhibition responses from a three arm vaccine trial (Mullen, et al, 2008).

Usage

```
data("ama1c1cpg")
```

Format

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

`vacc` a factor representing the three arms of the trial. The levels are: `20ug+CPG` `80ug` `80ug+CPG`

`resp` a numeric vector giving the response: day 70 sera percent in vitro growth inhibition of the 3D7 malaria parasite.

References

Mullen, GE, Ellis, RD, Miura, K, Malkin, E, Nolan, C, Han, M, Fay, MP, Saul, A, Zhu, D, Rausch, K, Moretz, S, Shou, H, Long, CA, Miller, LH, Treanor, J. 2008. Phase 1 trail of ama1-c1/alhydrogel plus cpg 7909: an asexual blood-stage vaccine for plasmodium falciparum malaria. PLoS ONE. 3(8):32940.

Examples

```
data(ama1c1cpg)
## maybe str(ama1c1cpg) ; plot(ama1c1cpg) ...
```

`anovaOneWay` *One-Way ANOVA*

Description

Do one-way ANOVA with estimates and confidence intervals on parameters. The parameter is called tau.sq and is the weighted sum of the square of the difference between the true means and the weighted average of the true means. Allows var.equal=FALSE using the Brown-Forsythe method that generalizes Welch's t-test to the k-sample problem.

Usage

```
anovaOneWay(y, g, var.equal = TRUE, nullValue = 0,
    parm =c("ICC","varb"), conf.level = 0.9)
```

Arguments

y	numeric vector of responses
g	group membership vector (may be numeric, character, or factor)
var.equal	logical, are the variances for all groups be equal? TRUE gives usual anova, FALSE gives Brown-Forsythe method.
nullValue	null value of tau.square (between group variance) or tau.sq/sigma.sq (must be 0 now)
parm	type of parameter, either 'ICC' (the parameter that R square estimates for this problem) or 'varb' (the between group variance).
conf.level	confidence level for the confidence interval. Default is 0.90 so that when the p-value<0.05, the two-sided confidence interval will exclude 0.

Details

The typical way to get the p-value for a one-way anova is `anova(lm(y~g))`. This function was written to add two new features.

First, using the method of Brown and Forsythe (1974a), the function allows for non-equal variances between the groups. This is one generalization of Welch's t-test to the one-way ANOVA case. Brown and Forsythe (1974b) give simulations showing that the type I error rate is close to the nominal (under the nomrality assumption with different variances).

Second, the function gives confidence intervals on either 'ICC' or 'varb'. The 'varb' (the between-group variance) is `sum((na/n)*(ua-u)^2)` where na is a vector of length k giving the sample size in each group, n is the total sample size, and ua is a vector of the k means in the groups, and u is the overall mean. Let varw be the within-group variance, then ICC=varb/(varb+varw). ICC is the intraclass correlation coefficient, and in this situation it is the parameter that the R square is estimating.

Value

A object of class 'htest'.

Note

Note also that it is possible to get a 90 pct confidence interval for varb that is (0,0). This occurs when the group means are much closer to each other than they would be expected to be by chance, given the observed variability between observations within the groups.

Author(s)

Michael P. Fay

References

Brown and Forsythe (1974a). Biometrics 30:719-724.

Brown and Forsythe (1974b). Technometrics 16: 129-132.

Examples

```
require(datasets)
library(asht)
ChickWeightTime20<-ChickWeight[ChickWeight$Time==20,]

anovaOneWay(1:10,c(rep(1,4),rep(2,6)))
anova(lm(weight~Diet,data=ChickWeightTime20))
t.test(ChickWeightTime20$weight[ChickWeightTime20$Diet==1],
       ChickWeightTime20$weight[ChickWeightTime20$Diet==2],
       var.equal=FALSE)
anovaOneWay(ChickWeightTime20$weight, ChickWeightTime20$Diet,
     var.equal=FALSE)
```

---

bfTest *Behrens-Fisher Test*

Description

Tests for a difference in means from two normally distributed variates with possibly different variances.

Usage

```
bfTest(x, ...)

## Default S3 method:
bfTest(x, y,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, conf.level = 0.95, control=bfControl(), ...)

## S3 method for class 'formula'
bfTest(formula, data, subset, na.action, ...)
```

Arguments

x	a (non-empty) numeric vector of data values.
y	an optional (non-empty) numeric vector of data values.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu	a number indicating the true value of the difference in means
conf.level	confidence level of the interval.
control	a list of arguments used for determining the calculation algorithm, see bfControl
formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.

data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further arguments to be passed to or from methods.

Details

Fisher (1935) developed a fiducial solution to the two-sample difference in means problem with normally distributed data with different variances. That has become known as the Behrens-Fisher solution. Robinson (1976) showed through extensive simulations, that the Behrens-Fisher solution is valid (i.e., the test gives type I error rate less than the significance level, and its confidence intervals on the difference in means have coverage at least as large as the nominal confidence level).

The following are the same as with the usual t-test in t.test. alternative = "greater" is the alternative that x has a larger mean than y. Missing values are silently removed. If the input data are effectively constant an error is generated.

Value

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

statistic	the value of the t-statistic.
parameter	R = atan(SEMx/SEMy) used in Behrens-Fisher distribution, where SEMx=std error of the mean of x, see pbf, but not used in calculation for this function
p.value	the p-value for the test.
conf.int	a confidence interval for the difference in means (mean.x-mean.y) appropriate to the specified alternative hypothesis.
estimate	the estimated means
null.value	the specified hypothesized value of the mean difference
alternative	a character string describing the alternative hypothesis.
method	a character string describing the test.
data.name	a character string giving the name(s) of the data.

References

Fisher, RA (1935). The fiducial argument in statistical inference. Annals of Eugenics. 6, 391-398.

Robinson, G (1976). Properties of Students t and of the Behrens-Fisher solution to the two means problem. The Annals of Statistics 4, 963-971 (Corr: 1982, p. 321).

See Also

The more common solution for this problem is Welch's t-test (the default in t.test). Welch's t-test does not guarantee that the type I error rate is less than the significance level, but it appears to work well in most cases.

Examples

```
## Classical example: Student's sleep data
## Traditional interface
with(sleep, bfTest(extra[group == 1], extra[group == 2]))
## Formula interface
bfTest(extra ~ group, data = sleep)
## Results are simular to Welch's t-test,
## but a little more conservative
t.test(extra~group,data=sleep)
```

`cvTest` *Coefficient of Variation Test*

Description

One-sample coefficient of variation tests and confidence intervals based on either normal or lognormal assumptions.

Usage

```
cvTest(x, nullCV = 1,
alternative = c("two.sided", "less", "greater"),
conf.level = 0.95, distn = c("normal", "lognormal"),
CVmax = 10^6)
```

Arguments

`x`	numeric vector
`nullCV`	null coefficient of variation, or CV on boundary between null and alternative hypotheses
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
`conf.level`	confidence level of the interval
`distn`	assumed distribution
`CVmax`	maximum coefficient of variation used in uniroot CI searches when distn='normal'

Value

A list of class 'htest'

`statistic`	mean
`parameter`	stadard deviation
`estimate`	estimate of coefficient of variation: sd(x)/mean(x) for distn='normal', and sqrt(exp(var(log(x)))-1) for distn='lognormal'
`p.value`	p.value associated with alternative

conf.int	confidence interval
null.value	null CV
alternative	alternative
method	description of method

Author(s)

Michael P. Fay

References

Koopmans, Owen, Rosenblatt (1964) "Confidence intervals for the coefficient of variation for the normal and log normal distributions" Biometrika 25-32.

Examples

```
cvTest(rnorm(25,mean=3,sd=.2),distn="normal")
```

---

meldCD *Meld Two Confidence Distributions*

Description

Melding is a very general way of combining two independent confidence interval proceedures to create a confidence interval on a function of the two associated parameters (e.g., difference or ratio).

Usage

```
meldCD(H1, H2, nullparm = NULL, parmtype = c("difference", "ratio", "oddsratio"),
    conf.level = 0.95, alternative = c("two.sided", "less", "greater"),
    estimate = c("median", "mean"), lim = c(-Inf, Inf), parmGrid = NULL,
    nmc = 1e5, ngrid = 1e4, calcmethod = "int", epsilon=1e-8, utol=1e-8)
```

Arguments

H1	a function representing the confidence distribution for parameter 1 (see details)
H2	a function representing the confidence distribution for parameter 2
nullparm	null parameter value for the parameter defined by parmtype
parmtype	parameter type, 'difference' gives parm2-parm1, 'ratio' gives parm2/parm1 (for 'oddsratio' see details).
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
conf.level	confidence level of the interval.
estimate	type of estimate derived from each confidence distribution, either 'median' or 'mean'

lim	a vector with limits on the parameters (both parameters should have the same limits)
parmGrid	a vector of a grid of possible values of the parameter, if NULL one is produced based on the lim argument
nmc	number of Monte Carlo replications, used if calcmethod='mc'
ngrid	minimum number of elements in the parameter grid, used if parmGrid=NULL
calcmethod	calculation method, either 'int' (numeric integration) or 'mc' (Monte Carlo)
epsilon	small value for warning check, we want the minimum of the CD over the parameter grid to be greater than epsilon, and the maximum to be less than 1-epsilon
utol	small value for passing to tol option in uniroot for confidence interval calculations

Details

For continuous responses, a confidence distribution (CD) is like a frequentist posterior distribution. We represent the CDs as cumulative distribution functions in the parameter space. The CD gets its name because it is created from the confidence interval process. If (L,U) is the 1-alpha confidence interval for group 1, then H1(L) = alpha/2 and H1(U)=1-alpha/2. Typically, the the CDs can be formulated as one-sided (alternative='greater') p-value functions, or 1-p for alternative='less', where the main function argument is the boundary value on the parameter space between the null and alternative. See binomial example below.

The median of the CD can be used as an estimate of the parameter.

We want inferences on a function of the parameters, say g(parm1, parm2), where when

- parmtype="difference" then g(parm1,parm2)=parm2-parm1
- parmtype="ratio" then g(parm1,parm2)=parm2/parm1
- parmtype="oddsratio" then g(parm1,parm2)=(parm2*(1-parm1))/(parm1*(1-parm2)).

The function g(parm1, parm2) must be increasing in parm2 and decreasing in parm1, so for example normal CDs (or any with a range -Inf to Inf) are not allowed for parmtype='ratio'. The `lim` argument checks to see if the parmtype is allowed.

Let T1 and T2 be simulated independent random variables associated with the CDs H1 and H2. Then to get a two-sided 1-alpha confidence interval on g(parm1,parm2) we can use `quantile(g(T1,T2),probs=c(alpha/2,` This is basically how it works when `calcmethod='mc'`. When `calcmethod='int'` then numeric integration is used.

For discrete responses, to ensure validity of the resulting confidence intervals, each group uses either a lower or upper CD, depending on the one-sided alternative. Thus, confidence intervals for two-sided alternatives cannot be calculated in one call to the `meldCD` for discrete data. See Fay, Proschan, and Brittain (2015) and the example.

Value

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

p.value	the p-value for the test.

conf.int	a confidence interval for the mean appropriate to the specified alternative hypothesis.
estimate	vector of parameter estimates for each group and using the parmtype, uses the median of the CDs for estimates
null.value	the specified hypothesized value of the difference in parameters
alternative	a character string describing the alternative hypothesis.
method	a character string describing the test.
data.name	a character string giving the name(s) of the data.

Warning

The function has not been tested for discrete confidence distributions. Note most confidence distributions for discrete data are not discrete CDs because the parameters are continuous.

Author(s)

Michael P. Fay

References

Fay, MP, Proschan, MA, Brittain, E (2015). Combining One-sample confidence procedures for inference in the two-sample case. Biometrics. 71: 146-156.

See Also

meldtTest and binomMeld.test for special cases.

Examples

```
x1<-4
n1<-11
x2<- 13
n2<-24


#  we use the upper and lower CDs
# this is needed for discrete data to ensure valid intervals
H1L<-function(theta){ pbeta(theta,x1,n1-x1+1)}
# Note, this is just a p-value function that inputs the null boundary value:
binom.test(x1,n1,p=.4,alternative="greater")$p.value
H1L(.4)
H1U<-function(theta){ pbeta(theta,x1+1,n1-x1)}
# Note, but this is just a function for 1-p that inputs the null boundary value:
1-binom.test(x1,n1,p=.4,alternative="less")$p.value
H1U(.4)
H2L<-function(theta){ pbeta(theta,x2,n2-x2+1)}
H2U<-function(theta){ pbeta(theta,x2+1,n2-x2)}

meldCD(H1U,H2L, lim=c(0,1),conf.level=0.975,alternative="greater")
meldCD(H1L,H2U, lim=c(0,1),conf.level=0.975,alternative="less")
```

```
# notice that the estimates are different than the usual
# difference in sample proportions
require(exact2x2)
binomMeld.test(x1,n1,x2,n2, conf.level=0.975, alternative="greater")
# compare to two-.sided from
binomMeld.test(x1,n1,x2,n2, conf.level=0.95, alternative="two.sided")
```

`meldtTest` *Meld t Test*

Description

Tests for a difference in parameters, when the parameter estimates are independent and both have t distributions.

Usage

```
meldtTest(x, y, alternative = c("two.sided", "less", "greater"), delta = 0,
    conf.level = 0.95, control = bfControl(), ...)
```

Arguments

`x`	a list from the first group with objects: estimate (estimate of parameter), stderr (standard error of the estimate), and df (degrees of freedom associated with t distribution)
`y`	a list from the second group with objects: estimate, stderr, and df
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`delta`	a number indicating the null hypothesis value of the difference in parameters when `alternative="two.sided"`. See details for one-sided hypotheses
`conf.level`	confidence level of the interval.
`control`	a list of arguments used for determining the calculation algorithm, see `bfControl`
`...`	further arguments to be passed to or from methods (currently not used)

Details

Suppose x$estimate and y$estimate estimate the parameters xParm and yParm. Let Delta=yParm-xParm. This function tests hypotheses of the form,

- alternative="two.sided" tests H0: Delta=delta versus H1: Delta != delta
- alternative="less" tests H0: Delta >= delta versus H1: Delta< delta
- alternative="greater" tests H0: Delta <= delta versus H1: Delta> delta

The test uses the theory of melding (Fay, Proschan and Brittain, 2015). The idea is to use confidence distribution random variables (CD-RVs). It is easiest to understand the melding confidence intervals by looking at the Monte Carlo implementation. Let nmc be the number of Monte Carlo replicates, then the simulated CD-RV associated with x are Bx = x$estimate + x$stderr * rt(nmc,df=x$df). Similarly define By. Then the 95 percent melded confidence interval for Delta=yParm-xParm is estimated by quantile(By-Bx, probs=c(0.025,0.975)).

When the estimates are means from normal distributions, then the meldtTest reduces to the Behrens-Fisher solution (see `bfTest`).

Only one of `x$stderr` or `y$stderr` may be zero.

Value

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

`statistic`	the value of the t-statistic.
`parameter`	R = `atan(x$stderr/y$stderr)` used in Behrens-Fisher distribution, see `pbf`
`p.value`	the p-value for the test.
`conf.int`	a confidence interval for the difference in means appropriate to the specified alternative hypothesis.
`estimate`	means and difference in means estimates
`null.value`	the specified hypothesized value of the difference in parameters
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string describing the test.
`data.name`	a character string giving the name(s) of the data.

Warning

If the two estimates are not independent, this function may give invalid p-values and confidence intervals!

Author(s)

Michael P. Fay

References

Fay, MP, Proschan, MA, Brittain, E (2015). Combining One-sample confidence procedures for inference in the two-sample case. Biometrics. 71: 146-156.

See Also

`bfTest` and `pbf`

Examples

```
## Classical example: Student's sleep data
## Compare to bfTest
xValues<- sleep$extra[sleep$group==1]
yValues<- sleep$extra[sleep$group==2]


x<-list(estimate=mean(xValues),
    stderr=sd(xValues)/sqrt(length(xValues)),
    df=length(xValues)-1)
y<-list(estimate=mean(yValues),
    stderr=sd(yValues)/sqrt(length(yValues)),
    df=length(yValues)-1)
bfTest(xValues,yValues)
# by convention the meldtTest does mean(y)-mean(x)
meldtTest(x,y)
meldtTest(y,x)
```

`metaNorm` *Meta analysis of normally distributed parameters with assumed known variance*

Description

Performs either a random effects meta analysis (Paule-Mandel method or Dersimonian-Laird method) or a fixed effects meta analysis.

Usage

```
metaNorm(y, s2, method = c("PM", "DL", "fixed"), df = NULL, nullparm = 0,
    conf.level = 0.95, alternative = c("two.sided", "less", "greater"),
    niter = 100, epsilon = 1e-10)
```

Arguments

`y`	vector of parameter estimates
`s2`	vector of variances of parameter estimates
`method`	either "PM" (Paule-Mandel random effects method), "DL" (Dersimonian-Laird random effects method) or "fixed" (fixed effects method)
`df`	degrees of freedom, NULL gives either df=k-1 (method="PM"), df=Inf (method="DL" or "fixed")
`conf.level`	confidence level
`alternative`	type of alternative hypothesis
`nullparm`	null value of the parameter for calculating the p-value
`niter`	maximum number of iterations for method="PM"
`epsilon`	small number for determining convergence of Paule-Mandel method.

Details

Assume you have a vector of treatment effect estimates from K studies (y), together with variance estimates (s2). Assume that y[i] is distributed normal with mean theta[i] and variance s2[i], and assume the theta[i] (the latent treatment effect for the ith study) is normally distributed with mean theta and variance tau2 (tau^2). Assume independence between studies.

We are interested in estimating the weighted average of the theta[i]. If tau2 is known, then an efficient estimator weighs each study proportional to the inverse of its variance, w[i] = 1/(tau2 + s2[i]). We can either assume tau2=0, and we have a fixed effects model (in other words, the treatment effect is constant across all the studies), or estimate tau2. The method for estimating tau2 either uses a simple method of moments estimator of Dersimonian and Laird (1986), or an iterative method of moments estimator of Paule and Mandel (1982). Dersimonian and Kacker (2007) give the details.

For the Paule-Mandel estimator, to account for the fact that we are estimating tau2, we default to using a t-distribution with K-1 degrees of freedom (for motivation see Brittain, Fay and Follmann, 2012, Supplement, Section 3).

Value

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

`statistic`	a vector of [1] the estimator of tau2 and [2] the t-statistic (or Z-statistic)
`parameter`	degrees of freedom of the t-distribution (df=Inf gives a normal distribution)
`p.value`	the p-value for the test.
`conf.int`	a confidence interval
`estimate`	a vector of [1] the estimated weighted means and [2] the estimated standard error of the weighted means
`null.value`	the specified hypothesized value of the weighted means
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string describing the test.
`data.name`	a character string giving the name(s) of the data.

Author(s)

Michael P. Fay

References

Brittain, Fay, and Follmann (2012) A valid formulation of the analysis of noninferiority trials under random effects meta-analysis. Biostatistics 13(4): 637-649.

Dersimonian, R and Kacker, R (2007) Random-effects model for meta-analysis of clinical trials: an update. Contemporary Clinical Trials 28:105-144.

Dersimonian, R and Laird, N. (1986). Meta-analysis in clinical trials. Controled Clinical Trials. 7:177-187.

Paule, RC and Mandel, J (1982). Consensus values and weighting factors. J Res Natl Bur Stand 87: 377-385.

See Also

meta package on CRAN

Examples

```
# Data from Table III of Teo et al, BMJ 303:1499-1503
# Effects of intravenous magnesium in suspected acute myocardial
# infarction: overview of randomised trials
# xt/nt = deaths/total in treatment group (magnesium)
# xc/nc = deaths/total in control group
xt<-c(1,9,2,1,10,1,1)
nt<-c(40,135,200,48,150,59,25)
xc<-c(2,23,7,1,8,9,3)
nc<-c(36,135,200,46,148,56,23)

rt<- xt/nt
rc<- xc/nc
logOR<- log(rt*(1-rc)/(rc*(1-rt)))
varLogOR<- 1/(nt*rt*(1-rt)) + 1/(nc*rc*(1-rc))

# Compare weighted mean and std err to Table 4 of Dersimonian and Kacker, 2007
metaNorm(logOR,varLogOR,method="PM")
metaNorm(logOR,varLogOR,method="DL")
metaNorm(logOR,varLogOR,method="fixed")
# Compare tau values to Table 3 of Dersimonian and Kacker, 2007
sqrt( metaNorm(logOR,varLogOR,method="PM")$statistic["tau squared"] )
sqrt( metaNorm(logOR,varLogOR,method="DL")$statistic["tau squared"] )
```

---

prevSeSp *Estimate prevalence with confidence interval accounting for sensitivity and specificity*

Description

Using the method of Lang and Reiczigel (2014), estimate prevalence and get a confidence interval adjusting for the sensitivity and specificity (including accounting for the variability of the sensitivity and specificity estimates).

Usage

```
prevSeSp(AP, nP, Se, nSe, Sp, nSp, conf.level = 0.95, neg.to.zero=TRUE)
```

Arguments

AP	apparent prevalence (proportion positive by test)
nP	number tested for AP
Se	estimated sensitivity (true positive rate)

nSe	number of positive controls used to estimate sensitivity
Sp	estimated specificity (1- false positive rate)
nSp	number of negative controls used to estimate specificity
conf.level	confidence level
neg.to.zero	logical, should negative prevalence estimates and lower confidence limits be set to zero?

Details

When measuring the prevalence of some disease in a population, it is useful to adjust for the fact that the test for the disease may not be perfect. We adjust the apparent prevalence (the proportion of people tested positive) for the sensitivity (true positive rate: proportion of the population that has the disease that tests positive) and the specificity (1-false positive rate: proportion of the population that do not have the disease that tests negative). So if the true prevalence is θ and the true sensitivity and specificity are Se and Sp, then the expected value of the apparent prevalence is the sum of the expected proportion of true positive results and the expected proportion of false positive results:

$$AP = \theta Se + (1 - Sp)(1 - \theta).$$

Plugging in the estimates (and using the same notation for the estimates as the true values) and solving for θ we get the estimate of prevalence of

$$\theta = \frac{AP - (1 - Sp)}{Se - (1 - Sp)}.$$

Lang and Reiczigel (2014) developed an approximate confidence interval for the prevalence that not only adjusts for the sensitivity and specificity, but also adjusts for the fact that the sensitivity is estimated from a sample of true positive individuals (nSe) and the specificity is estimate from a sample of true negative individuals (nSp).

If the estimated false positive rate (1-specificity) is larger than the apparent prevalence, the prevalence estimate will be negative. This occurs because we observe a smaller proportion of positive results than we would expect from a population known not to have the disease. The lower confidence limit can also be negative because of the variability in the specificity estimate. The default with neg.to.zero=TRUE sets those negative estimates and lower confidence limits to zero.

The Lang-Reiczigel method uses an idea discussed in Agresti and Coull (1998) to get approximate confidence intervals. For 95% confidence intervals, the idea is similar to adding 2 positive and 2 negative individuals to the apparent prevalence results, and adding 1 positive and 1 negative individual to the sensitivity and specificity test results, then using asymptotic normality. Simulations in Lang and Reiczigel (2014) show the method works well for true sensitivity and specificity each in ranges from 70% to over 90%.

Value

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

estimate	the adjusted prevalence estimate, adjusted for sensitivity and specificity
statistic	the estimated sensitivity given by Se

parameter	the estimated specificity given by Sp
conf.int	a confidence interval for the prevalence.
method	the character string describing the output.
data.name	a character string giving the unadjusted prevalence value and the sample size used to estimate it (nP).

Note

There is a typo in equation 4 of Lang and Reiczigel (2014), the $(1 + \hat{P})^2$ should be $(1 - \hat{P})^2$.

Author(s)

Michael P. Fay

References

Agresti, A., Coull, B.A., 1998. Approximate is better than 'exact' for interval estimation of binomial proportions. Am. Stat. 52,119-126.

Lang, Z. and Reiczigel, J., 2014. Confidence limits for prevalence of disease adjusted for estimated sensitivity and specificity. Preventive veterinary medicine, 113(1), pp.13-22.

See Also

truePrev in package **prevalence** for Bayesian methods for this problem (but this requires JAGS (Just Another Gibbs Sampler), a separate software that can be called from R if it is installed on the user's system.)

Examples

```
# Example 1 of Lang and Reiczigel, 2014
# 95% CI should be 0.349, 0.372
prevSeSp(AP=4060/11284,nP=11284,Se=178/179,nSe=179,Sp=358/359, nSp=359)

# Example 2 of Lang and Reiczigel, 2014
# 95% CI should be  0, 0.053
prevSeSp(AP=51/2971,nP=2971,Se=32/33,nSe=33,Sp=20/20, nSp=20)

# Example 3 of Lang and Reiczigel, 2014
# 95% CI should be 0 and 0.147
prevSeSp(AP=0.06,nP=11862,Se=0.80,nSe=10,Sp=1, nSp=12)

# Example 4 of Lang and Reiczigel, 2014
# 95% CI should be 0.58 to 0.87
prevSeSp(AP=259/509,nP=509,Se=84/127,nSe=127,Sp=96/109, nSp=109)
# 95% CI should be 0.037 to 0.195
prevSeSp(AP=51/509,nP=509,Se=23/41,nSe=41,Sp=187/195, nSp=195)
```

quantileTest *Tests and Confidence Intervals about a Quantile.*

Description

The ath quantile of a distribution is the value, q, such that F(q-) <= a <= F(q), where F(x)=Pr[X <= x]. These are exact tests and confidence intervals on independent observations that do not any assumptions on the distribution, F. For example, the tests are exact when data are discrete or continuous, and when the distribution is non-symmetric.

Usage

```
## S3 method for class 'ordered'
quantileTest(x,...)

## Default S3 method:
quantileTest(x, q = 0, prob = 0.5,
   alternative = c("two.sided", "less", "greater"),
   conf.level = 0.95, ...)

medianTest(x, m=0, ...)
```

Arguments

x	a vector of numeric, integer or ordered factor values
q	null quantile for test
m	null median for test
prob	quantile
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
conf.level	confidence level of the interval
...	further arguments to be passed to or from methods.

Details

A test on the quantile. The medianTest is just a wrapper function to call quantileTest with prob=0.5.

Ordinal factors may be used. The calculations just use as.numeric(x) for the factors, then return the character associated with that value. Estimates that are between two ordered factors, say "C" and "D", return the character "C/D".

Value

A list of class 'htest'.

Author(s)

Michael P. Fay

See Also

signTest

Examples

```
## For Poisson(mean=2.5) the median is 2
x<-rpois(20,2.5)
medianTest(x)
x<-ordered(c(rep("A",10),rep("B",60),rep("C",30)),levels=c("A","B","C"))
xnum<-as.numeric(x)
quantileTest(xnum,q=2,prob=0.705)
quantileTest(x,q=2,prob=0.705)
```

---

signTest *Exact Sign Test with Confidence Intervals*

Description

Uses link{binom.exact} or mcnemarExactDP to create sign test with confidence intervals on different parameters. Mid-p versions are available for some parameterizations (see details).

Usage

```
signTest(x, stat=c("cd","cpp","ud"), nullparm=NULL,
   alternative=c("two.sided","less","greater"), conf.level=0.95,...)
```

Arguments

x	numeric vector
stat	statistic for estimates and confidence intervals, "cd"= conditional difference: proportion positive - proportion negative, "cpp"= conditional proportion positive, and "ud"= unconditional difference: proportion positive-proportion negative (conditional proportions are out of non-zero values, unconditional are out of all values)
nullparm	null parameter value associated with stat, NULL value defaults to the exact sign test (i.e., stat="cd" and codestat="ud" gives 0, and stat="cpp" gives 0.5).
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
conf.level	confidence level of the interval
...	arguments passed to binom.exact or mcnemarExactDP

Details

The sign test is a conditional test, conditioning on the total number of non-zero observations testing that the proportion positive is different (or less, or greater) than the proportion negative. When the responses are differences in paired binary observations this is the same as a McNemar test.

This function gives estimates and confidence intervals compatible with the exact sign test for three different parameterizations. Let n.pos, n.neg,

and n.nonzero

be the number of positive, negative, and non-zero observations respectively out of `n=length(x)`. The conditional proportion positive are n.pos/n.nonzero, and the unconditional proportion positive are n.pos/n. Similarly, the conditional proportion negative are n.neg/n.nonzero and the unconditional proportion negative are n.neg/n. When `stat='cd'` the parameterization is the conditional difference in proportions (pos-neg), and when `stat='ud'` the parameterization is the uncondtional difference in proportions (pos-neg). The third parameterization is `stat='cpp'` the conditional proportion positive. The argument `nullparm` gives the null value of the test when `alternative='two.sided'`. When `nullparm=NULL`, this gives the traditional sign test, where `nullparm=0` for `stat='cd'` and `stat='ud'` and `nullparm=0.5` for `stat='cpp'`.

Conditioning on m=n.nonzero, Y is binomial with parameters m and beta. So when `stat='cpp'` the parameter we are estimating is beta, and when `stat='cd'` the parameter we are estimating is beta - (1-beta) = 2*beta-1. We use `binom.exact` to do the p-value and confidence interval calculations. Thus, `midp` versions and different two-sided methods (given by `tsmethod`) can be calculated.

Unconditionally, we treat M (the number non-zero) as a random variable, and assume M is binomial with parameters n and theta. When `stat='ud'` the parameter we are estimating is delta = theta*(2*beta-1), which is the unconditional difference: (proportion positive out of the total) - (proprtion negative out of the total). We use `mcnemarExactDP` to do the the p-value and confidence interval calculations. The methods associated with that function are described in Fay and Lumbard (2020). As of now, when `stat='ud'` a `midp` version is not available, and the only two-sided method available is a 'central' one, meaning the error for the 95% confidence interval is bounded by 2.5% on each side.

Value

A list of class 'htest' (use `str` to see elements)

`statistic`	vector of number of positive, negative, zero, and non-zero
`estimate`	vector of estimates related to `stat` argument
`p.value`	p.value associated with alternative
`conf.int`	confidence interval
`null.value`	null parameter value
`alternative`	alternative
`method`	description of method
`data.name`	name of x argument

Note

The sign test can be interpreted as a test that the median is zero assuming continuous data. If you want to test on the median without making continuity assumptions use `medianTest`.

Previous versions of `signTest` had `stat='pos-neg'` and `stat='prop pos'`, which are now referred to as `stat='cd'` and `stat='cpp'`, respectively. The old names give a warning, but may be removed in future versions.

Author(s)

Michael P. Fay

References

Fay MP, and Lumbard, K (2020). Confidence Intervals for Difference in Proportions for Matched Pairs Compatible with Exact McNemar's or Sign Tests. (unpublished manuscript).

Examples

```
x<-c(rep(-1,10),rep(0,60),rep(1,30))
signTest(x, stat='cd')
signTest(x, stat='cpp')
signTest(x, stat='ud')
# sample median is zero,
# and not surprisingly the median test
# properly gives a large p-value
medianTest(x)
```

`simulateSS` *Simulate sample sizes*

Description

A function that simulates sample sizes in an efficient manner. Inputs two functions: (1) a decision function that returns 1=reject, or 0=fail to reject, and (2) a data generating function.

Usage

```
simulateSS(decFunc, dataGenFunc, nstart = 100, numBatches = 100, repsPerBatch = 100,
   power = 0.3, alpha = 0.025, nrepeatSwitch = 3, printSteps = TRUE)
```

Arguments

`decFunc`	decision function, inputs data from dataGenFunc and outputs 1 (reject) or 0 (fail to reject).
`dataGenFunc`	data generating function, inputs a sample size and outputs simulated data object. Class of the data must match input for decFunc.
`nstart`	starting sample size value

numBatches	number of batches (must be at least 5), default=100
repsPerBatch	number of replications per batch (must be at least 10), default=100
power	power desired
alpha	one-sided alpha level, used for estimating power from batches by normal approximation
nrepeatSwitch	one of 2,3,4 or 5. default=3. If nrepeatSwitch batch estimates of sample size are the same in a row, then switch to an up-and-down method (adding or subtracting 1 to sample size).
printSteps	logical, print intermediate steps of algorithm?

Details

This is an algorithm proposed in Fay and Brittain (2022, Chapter 20). Here are the details of the algorithm. For step 1, we pick a starting sample size, say N_1, and the number of replications within a batch, m, and the total number of batches, b_tot. We simulate m data sets with sample size N_1, and get the proportion of rejections, say P_1. Then we use a normal approximation to estimate the target sample size, say N_norm. In step 2, we replicate m data sets with sample size $N_2 = N_norm$ to get the associated proportion of rejections, say P_2. We repeat 2 more batches with $N_3=N_norm/2$ and $N_4=2 N_norm$, to get proportions P_3, and P_4. Then in step 3, we use isotonic regression (which forces monotonicity, power to be non-decreasing with sample size) on the 4 observed pairs ($(N_1,P_1),\ldots,(N_4,P_4)$), and linear interpolation to get our best estimate of the sample size at the target power, N_target. We use that estimate of N_target for our sample size for the next batch of simulations. This idea is of using the best estimate of the target for the next iteration is studied in Wu (1985, see Section 3). Step 4 is iterative, for the ith batch we repeat the isotonic regression, except now with N_i estimated from the first $(i-1)$ observation pairs. We repeat step 4 until either the number of batches is b_tot, or the current sample size estimate is the same as the last nrepeatSwitch-1 estimates, in which case we switch to an up-and-down-like method. For each iteration of the up-and-down-like method, if the current proportion of rejections from the last batch of m replicates is greater than the target power, then subtract 1 from the current sample size estimate, otherwise add 1. Continue with that up-and-down-like method until we reach the number of batches equal to b_tot. The up-and-down-like method was added because sometimes the algorithm would get stuck in too large of a sample size estimate.

Value

A list with elements:

N	vector of estimated sample sizes at the end of each batch
P	vector of power estimates at the end of each batch
Nstar	estimate of sample size, not necessarily an integer
Nestimate	integer estimate of sample size equal to ceiling(Nstar)

Author(s)

Michael P. Fay

References

Fay, M.P. and Brittain, E.H. (2022). Statistical Hypothesis Testing in Context. Cambridge University Press. New York.

Wu, CJ (1985). Efficient sequential designs with binary data. Journal of the American Statistical Association. 19: 1085-1098.

Examples

```
# simple example to show method
# simulate 2-sample t-test power
# for this simple case, better to just use power.t.test
power.t.test(delta=.5,sig.level=0.025,power=.8,
   type="two.sample",alternative="one.sided")
decFunc<-function(d){
  ifelse(t.test(d$y1,d$y2,alternative="less")$p.value<=0.025,1,0)
}
dataGenFunc<-function(n){
  list(y1=rnorm(n,0),y2=rnorm(n,.5))
}
# for example use on 20 batches with 20 per batch
set.seed(1)
simulateSS(decFunc,dataGenFunc,nstart=100,numBatches=100,repsPerBatch=100,
   power=0.80, alpha=0.025,printSteps=FALSE)
```

`tukeyWelsch` *Tukey-Welsch Pairwise Tests*

Description

Calculate pairwise comparisons between groups levels using step down correction for multiple testing.

Usage

```
tukeyWelsch(y, g, method = c("aov", "kw", "sr", "user"),
    pvalfunc = NULL, padjfunc = padjTW, maxnTest=10^4,nTestMessage=FALSE, ...)
```

Arguments

`y`	response vector
`g`	grouping vector or factor
`method`	type of method for tests, one of 'aov' (ANOVA which is a t-test for the pairwise comparisons) 'kw' (Kruskal-Wallis test, which is a Wilcoxon-Mann-Whitney test for the pairwise comparisons), 'sr' (studentized range test), or 'user' (user supplied function, see details).
`pvalfunc`	function to test for effects and return a p-value. Used if method='user'. See details.

padjfunc	function that takes the unadjusted p-value vector from a stage, and returns the adjusted p-value vector for that stage (see details)
maxnTest	maximum number of tests, if the number of tests is larger than maxnTest then gives an error
nTestMessage	logical, print a message at the start of calculations telling how many tests will be calculated
...	additional arguments to pass to XXX (if method='aov'), YYY (if method='kw') or pvalfunc (if method='user')

Details

This function does a k-sample test (either one-way ANOVA [method='aov'] or Kruskal-Wallis test [method='kw']) on the responses when the g vector has k levels. Then it does all the pairwise comparisons (either t.tests [method='aov'] or Wilcoxon-Mann-Whitney tests [method='kw']) giving multiple comparison adjusted p-values. The adjustment uses a step-down method, that is different from (and potentially more powerful than) the single step procedures in pairwise.t.test and pairwise.wilcox.test. The method is described in Einot and Gabriel (1975) [for the anova case] and Campbell and Skillings (1985) for the Kruskal-Wallis case. See also Hochberg and Tamhane (1987, p. 111 for 'aov' case, and p. 247-248 for the 'kw' case).

Here are the details. First, the k-sample test is done, where the type of test is determined by the method. The function repeats that type of test k-1 times, leaving out a different level of the group each time. These are k-1 tests, each having k-1 levels. This process repeats itself (i.e., do choose(k,k-2) tests each having k-2 levels, then do choose(k,k-3) tests each having k-3 levels, etc) until we get to the choose(k,2) pairwise tests. Reject at level aj = 1- (1-alpha)^(j/k), for all tests where there are j groups, for j=2,..,k-2 and at level aj=alpha for j=k-1 and k. These adjusted significance levels are known as the Tukey-Welch (see Hochberg and Tamhane, p. 111) or Ryan (see Einot and Gabriel, 1975) levels. Then we only reject each pairwise comparison, if we reject at all null hypotheses that contain that pair.

We convert this procedure into adjusted p-values, by finding the lowest alpha level such that each pairwise comparison would be rejected, that is its adjusted p-value. The padjfunc is a function that takes the unadjusted p-values and gives the adjustment for each level by itself. For example, the default uses the Tukey-Welch adjusted significance levels, and the function solves for alpha as a function of aj (i.e., inputs unadjP and returns either 1-(1-unadjP^(k/j) for j=2,3,...k-2 or unadjP for j=k-1 or k)). Then taking the individual level adjusted p-values, we define the step-down adjusted p-value for each pairwise comparison as the maximum of all the individual level adjusted p-values for each hypothesis that contains the pair as part of its groups tested.

When k=3, this method gives an adjusted p-value for each pairwise comparison that is the maximum of the k-sample test p-value and the unadjusted p-value for the two-sample test using that pair of levels.

When method='user' the function uses the pvalfunc function to test for p-values. The function must input y and g and output the p-value for the j-sample test, where j is the number of levels present in g. So the function must be defined when j=2,3,...,k.

Value

An object of class 'tukeyWelsch', a list with elements:

`fullResults`	a list of all the intermediate p-values (unadjusted and adjusted). Not printed by default
`method`	description of method
`data.name`	description of input data
`ksample.pvalue`	p-value for k-sample test
`pairwise.pvalues`	vector of adjusted p-values for pairwise comparisons

Author(s)

Michael P. Fay

References

Campbell and Skillings (1985) JASA 998-1003.

Einot and Gabriel (1975) JASA 574-583.

Hochberg, Y and Tamhane, AC (1987) Multiple Comparison Procedures. Wiley: New York.

See Also

`pairwise.wilcox.test` and `pairwise.t.test`

Examples

```
##
createData<-function(n,props,shifts,ry=rnorm){
  k<-length(props)
  if (round(sum(props),8)!=1) stop("sum of props must be 1")
  props<- props/sum(props)
  if (length(shifts)!=k) stop("length of shifts must equal length of props")
  g<-rep(1:k,as.vector(rmultinom(1,n,props)))
  y<-ry(n)
  for (i in 1:k){
    y[g==i]<-y[g==i]+shifts[i]
    }
  list(y=y,g=g)
}
set.seed(1)
d<-createData(100,c(.2,.3,.2,.3),c(0,0,0,1))
tukeyWelsch(d$y,factor(d$g),method="kw")
tukeyWelsch(d$y,factor(d$g),method="aov")
tukeyWelsch(d$y,factor(d$g),method="sr")
TukeyHSD(aov(d$y~factor(d$g)))[[1]][,"p adj"]
```

var1Test *One Sample Test of Normal Variance*

Description

Give tests and confidence intervals on the variance of a sample from a normal distribution.

Usage

```
var1Test(x, nullVar = 1,
   alternative = c("two.sided", "less", "greater"),
   conf.level = 0.95)
```

Arguments

x	numeric vector
nullVar	null variance, or variance on the boundary between the null and alternative hypotheses
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
conf.level	confidence level of the interval

Details

Tests derived from normality assumption.

Value

A list of class 'htest' (use str to see elements)

Author(s)

Michael P. Fay

References

Dudewicz, EJ and Mishra, SN (1988) Modern Mathematical Statistics. Wiley. (Section 9.6).

Examples

```
var1Test(rnorm(25))
```

`wmwTest` *Wilcoxon-Mann-Whitney test with Confidence Interval on Mann-Whitney Parameter*

Description

The `wmwTest` function calculates the Wilcoxon-Mann-Whitney test (normal approximation, exact complete enumeration, and exact Mante Carlo implementation) together with confidence intervals on the Mann-Whitney parameter, Pr[X<Y] + 0.5 Pr[X=Y].

Usage

```
wmwTest(x, ...)

## Default S3 method:
wmwTest(x, y, alternative = c("two.sided", "less", "greater"),
  phiNull = 0.5, exact = NULL, correct = TRUE, conf.int = TRUE, conf.level = 0.95,
   latentContinuous = FALSE, method = NULL, methodRule = methodRuleWMW,
   tsmethod = c("central", "abs"), control = wmwControl(),...)

## S3 method for class 'formula'
wmwTest(formula, data, subset, na.action, ...)

## S3 method for class 'matrix'
wmwTest(x,...)
```

Arguments

`x`	a (non-empty) numeric vector of data values from group 1, or a contingency table matrix with 2 rows with the top row representing group 1 and the bottom row group 2
`y`	an optional (non-empty) numeric vector of data values from group 2
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`phiNull`	null hypothesis value for the Mann-Whitney parameter, Pr[X<Y]+0.5*Pr[X=Y]. Defaults to 0.5.
`exact`	logical, should exact test be calculated? (see method)
`correct`	a logical indicating whether to apply continuity correction in the normal approximation for the p-value and confidence interval (when method='asymptotic')
`conf.int`	logical, should confidence intervals be calculated?
`conf.level`	confidence level of the interval.

latentContinuous
: logical, should estimates and confidence intervals be presented as latent continuous parameters? (see details)

method
: character defining method, one of 'asymptotic', 'exact.ce' (exact by complete enumeration), 'exact.mc' (exact by Monte Carlo approximation). NULL value defaults to result of methodRule

methodRule
: function that inputs x,y, and exact and outputs a method, see methodRuleWMW

tsmethod
: two-sided method, either 'central' (double the one-sided p-values) or 'abs' (test statistic uses absolute value of difference in phi estimate and phiNull)

control
: a list of arguments for control of algorithms and output, see wmwControl

formula
: a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.

data
: an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).

subset
: an optional vector specifying a subset of observations to be used.

na.action
: a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").

...
: further arguments to be passed to or from methods.

Details

The function wmwTest evaluates the Wilcoxon-Mann-Whitney test (also called the Mann-Whitney U test or the Wilcoxon rank sum test). The WMW test is a permutation two-sample rank test, and the test may be evaluated under many different sets of assumptions (Fay and Proschan, 2010). The least restrictive set of assumptions tests the null hypothesis that the two distributions of the two samples are equal versus the alternative that they are different. Unfortunately, with only those assumptions, we cannot get confidence intervals on the Mann-Whitney parameter, phi= Pr[X<Y] + 0.5 Pr[X=Y]. In order to get confidence intervals on phi, we need additional assumptions, and for this function we use the proportional odds assumption. This assumption can be interpreted as saying that there exists some unknown monotonic transformation of the responses that leads to a location shift in a logistic distribution. This can work for discrete data (i.e., with ties allowed) if we interpret discrete responses as a grouping of some underlying latent continuous response. The proportional odds assumption is less restrictive that the assumption used in wilcox.test, which assumes a location shift on the unknown continuous distribution of the untransformed data.

In summary, the two-sided p-value can be interpreted as testing the null that the two distributions are equal, and the confidence intervals on the Mann-Whitney parameter are intrepreted under the proportional odds assumption. In general the confidence intervals are compatible with the associated p-values, for details see Fay and Malinovsky (2018).

There is a choice of three methods. When method='asymptotic', the test is implemented using a normal approximation, with (correct=TRUE) or without (correct=FALSE) a continuity correction. The resulting p-values should match wilcox.test (when paired=FALSE and exact=FALSE). When method='exact.ce', the test is implemented using complete enumeration of all permutations, and hence is only tractible for very small sample sizes (less than 10 in each group). When method='exact.mc', the test is implemented using Monte Carlo with B=10^4 replications (change

B with `control=controlWMW(nMC=B)`). As B gets larger the p-value approaches the exact one. (See 'note' section, sometimes the `method='exact.mc'` will not work.)

The `tsmethod='central'` gives two-sided p-value that is equal to `min(1,min(2*pless,2*pgreater))`. Alternatively, `tsmethod='abs'` gives the two-sided method, which is based on the test statistic |phi - phiNull|. Under the proportional odds assumption, `tsmethod='central'` allows us to interpret p.value/2 as one-sided p-values (this is not allowed using `tsmethod='abs'`). With continuous data, the p-values will be the same, but with ties they can be different.

From the two groups x (or top row of contringency table, or first factor in rhs of formula) and y (or bottom row of contingency table, or second factor in rhs of formula) the Mann-Whitney parameter represents Pr[X<Y]+0.5Pr[X=Y]. It is also the area under the curve of an ROC curve (see Hanley and McNeil, 1982). The confidence interval when `method='asymptotic'` generalizes the Method 5 of Newcombe (2006), which was a score-type modification of the interval of Hanley and McNeil (1982). The generalization is that the confidence interval adjusts for ties and allows a continuity correction (see examples below).

The `methodRule` function allows automatic choice of the method of calculation based on the data and the `exact` argument.

When the data are discrete, we can treat the data as if they are a grouping of some underlying continuous responses. Using the proportional odds assumption, we can then translate the Mann-Whitney parameter on the observed discrete data into the Mann-Whitney parameter on the latent continuous data (when `latentContinuous=TRUE` and using the default `control=controlWMW(latentOutput='mw')`). You can also translate the results into the proportional odds parameter on the latent continuous responses (when `latentContinuous=TRUE` and using `control=controlWMW(latentOutput='po')`). Translation is done with `latentTransform`.

Value

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

`statistic`	U statistic estimate of the Mann-Whitney parameter.
`parameter`	tie factor
`p.value`	the p-value for the test.
`conf.int`	a confidence interval for the Mann-Whitney parameter appropriate to the specified alternative hypothesis.
`estimate`	the estimated difference in means
`null.value`	the specified hypothesized value of the mean difference
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string describing the test.
`data.name`	a character string giving the name(s) of the data.

Warning

The algorithm for calculating the confidence interval when `tsmethod='abs'` is not guaranteed to give the correct value. It is possible to skip over a value. For more accurate results increase `control=wmwControl(rcheckgrid)` and `control=wmwControl(ncheckgrid)`

Note

The `method='exact.mc'` can sometimes fail. The issue is that for some Monte Carlo simulations the one-sided p-value function is not monotonic, even in for data sets where the one-sided p-value would be monotonic if we could do complete enumeration. In this case, the confidence limit will be set to NA and a warning will suggest trying `method='asymptotic'` or `method='exact.ce'` if feasible. Here is an example where that occurs: `set.seed(1); g<- c(rep(0,6),1,rep(0,4),1,rep(0,3),1,1,0,1,1,0,rep(1`
`y<-1:26; wmwTest(y~g,exact=TRUE)`.

References

Fay, MP and Malinovsky, Y (2018). Confidence Intervals of the Mann-Whitney Parameter that are Compatible with the Wilcoxon-Mann-Whitney Test. Statistics in Medicine: DOI: 10.1002/sim.7890.

Fay, MP and Proschan MA (2010). Wilcoxon-Mann-Whitney of t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. Statistics Surveys 4:1-39.

Hanley, JA, and McNeil, BJ (1982). The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. Radiology 143: 29-36.

Newcombe, Robert G. (2006). Confidence intervals for an effect size measure based on the Mann-Whitney statistic. Part 2: asymptotic methods and evaluation. Statistics in medicine 25(4): 559-573.

See Also

See `wilcox.test` for either exact p-value or the same asymptotic p-value and confidence interval on location shift under the shift assumption.

See `wilcox_test` for exact p-value and exact confidence interval on location shift.

Examples

```
# data from Table 1 of Hanley and McNeil (also given in Table 1 of Newcombe, 2006)
HMdata<-matrix(c(33,3,6,2,6,2,11,11,2,33),nrow=2,dimnames=
   list(c("Normal","Abnormal"),
   c("Definitely Normal",
   "Probably Normal",
   "Questionable",
   "Probably Abnormal",
   "Definitely Abnormal")))
HMdata
# to match Newcombe (2006, Table 1, Method 5) exactly
# use correct=FALSE and RemoveTeAdjustment=TRUE
wmwTest(HMdata, correct=FALSE, RemoveTieAdjustment=TRUE)
# generally smaller intervals with closer to nominal coverage with
# tie adjustment and continuity correction
wmwTest(HMdata)
```

WprevSeSp *Weighted prevalence inferences adjusted for sensitivity and specificity*

Description

Prevalence inferences from weighted survey data adjusted for sensitivity and specificity estimates as well as their variability.

Usage

```
WprevSeSp(
  x,
  n,
  w,
  cn,
  mn,
  cp,
  mp,
  method = c("binomial", "poisson"),
  conf.level = 0.95,
  nmc = 1e+05,
  seed = 49201
)
```

Arguments

x	integer vector of apparent positive tests for each group
n	integer vector of number of tests for each group
w	numeric vector of weights for each group (must sum to 1)
cn	number of positive tests for negative controls
mn	number of negative controls tested
cp	number of positive tests for positive controls
mp	number of positive controls tested
method	either "binomial" or "poisson"
conf.level	confidence level of the interval
nmc	number of Monte Carlo replications
seed	seed for random number generation

Details

This function is for estimating prevalence from a population using a complex survey with an imperfect assay. The survey data are partitioned into K strata (or groups), and the ith stratum is weighted by w[i] and we observe a proportion x[i]/n[i] that test positive on the assay. The weights are required to sum to 1, so that the apparent prevalence (i.e., the prevalence assuming the assay is perfect) is a

weighted average of the proportions of the strata (i.e., `sum(w*(x/n))`). The assay's sensitivity and specificity is measured by two populations, a positive control population gives a sensitivity estimate (`cp/mp`), and a negative control population gives a specificity estimate (`1 - cn/mn`). The adjusted prevalence estimate (adjusted for sensitivity and specificity) is the standard one described by Rogan and Gladen (1978). The `WprevSeSP` gives confidence intervals on the prevalence that account for the sampling variability in the survey (by a multinomial approximation), as well as for the varibility in the sensitivity and specificity estimates, but it does not account for the variability of the weights.

There are two methods, the 'binomial' method is based on the method of Korn and Graubard (1998) which is developed using a modification of exact binomial intervals, and the 'poisson' method which is based on the gamma method of Fay and Feuer (1997) which is developed assuming a weighted sum of Poisson variates. Simulations show that the 'poisson' method had greater than nominal coverage in all simulated scenarios at the cost of conservativeness, while the 'binomial' method was less conservative but had less than nominal coverage in some scenarios.

For details see Bayer, Fay and Graubard (2023).

Value

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

`estimate`	the adjusted prevalence estimate, adjusted for sensitivity and specificity
`statistic`	the estimated sensitivity given by cp / mp
`parameter`	the estimated specificity given by 1 - cn/ mn
`conf.int`	a confidence interval for the prevalence
`data.name`	a character string giving the unadjusted prevalence value
`method`	the character string describing the output

References

Bayer, D.M., Fay, M.P., and Graubard, B.I. (2023). "Confidence intervals for prevalence estimates from complex surveys with imperfect assays" arXiv:2205.13494.

Fay MP, Feuer EJ. Confidence intervals for directly standardized rates: a method based on the gamma distribution. Statistics in Medicine 1997; 16(7): 791-801.

Korn EL, Graubard BI. Confidence intervals for proportions with small expected number of positive counts estimated from survey data. Survey Methodology 1998; 24: 193-201.

Rogan WJ, Gladen B. Estimating prevalence from the results of a screening test. Am J Epidemiol 1978; 107(1): 71-76.

Examples

```
example_data_WprevSeSp <- list(
  x = c(
    53, 47, 63, 50, 54, 54, 57, 51, 66, 51, 52, 48, 37, 44, 59,
    55, 50, 58, 52, 54, 41, 45, 49, 54, 37, 53, 57, 58, 55, 55, 56,
    42, 58, 47, 49, 63, 54, 54, 54, 41, 43, 56, 44, 49, 47, 45, 62,
    53, 54, 47
  ),
  n = c(
```

```
    200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200,
    200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200,
    200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200,
    200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200, 200
  ),
  w = c(
    0.0205962892844504, 0.0204062236737538, 0.0203843096373626,
    0.0202785701233134, 0.0202617051778543, 0.0202138087214499, 0.0201972974884707,
    0.0201818190015587, 0.0201631543739836, 0.0201560795402158, 0.0201555234250465,
    0.0201461978246263, 0.0201342022821394, 0.0201264004067009, 0.0201167314250592,
    0.0201015081093692, 0.0201003484427457, 0.0201002680000886, 0.0200817537259523,
    0.0200573433887284, 0.0200443907258367, 0.0200358187073312, 0.0200349749335002,
    0.0200264994605187, 0.0200112846914561, 0.020006219121804, 0.0199975642569458,
    0.0199649774153205, 0.0199614929059539, 0.0199426355876479, 0.0199334287088002,
    0.0199298633246975, 0.0199150015155486, 0.0199063452368827, 0.0198920051366782,
    0.0198877425787182, 0.0198679831412633, 0.0198500844815989, 0.0198381388412286,
    0.0198348595904904, 0.0198348180141822, 0.0198174510243331, 0.0197922036364436,
    0.0197821574067888, 0.0197204417557631, 0.0197004976818864, 0.019682896458092,
    0.019649677766428, 0.0196158425485035, 0.019563169292488
  ),
  cn = 77,
  cp = 58,
  mn = 300,
  mp = 60
)


WprevSeSp(
  method = "binomial",
  x = example_data_WprevSeSp$x,
  n = example_data_WprevSeSp$n,
  w = example_data_WprevSeSp$w,
  cn = example_data_WprevSeSp$cn,
  mn = example_data_WprevSeSp$mn,
  cp = example_data_WprevSeSp$cp,
  mp = example_data_WprevSeSp$mp
)

WprevSeSp(
  method = "poisson",
  x = example_data_WprevSeSp$x,
  n = example_data_WprevSeSp$n,
  w = example_data_WprevSeSp$w,
  cn = example_data_WprevSeSp$cn,
  mn = example_data_WprevSeSp$mn,
  cp = example_data_WprevSeSp$cp,
  mp = example_data_WprevSeSp$mp
)
```

`wspoissonTest` *Test and Confidence Intervals on Weighted Sum of Poissons*

Description

The test is not as important as the confidence intervals, which are often used for directly standardized rates. The default uses the gamma method by fay and Feuer (1997), which by all simulations appears to retain nominal coverage for any set of parameters or weights. There is a mid-p-like version that is less conservative.

Usage

```
wspoissonTest(x, w, nullValue = NULL,
   alternative = c("two.sided", "less", "greater"),
   conf.level = 0.95, midp = FALSE, nmc = 0,
   wmtype = c("max", "mean", "minmaxavg", "tcz"),
   mult = 1, unirootTolFactor=10^(-6))
```

Arguments

x	a vector of counts (each assumed Poisson with a different parameter)
w	a vector of weights.
nullValue	a null hypothesis value of the weighted sum of the Poisson means, if NULL no test is done.
alternative	type of alternative hypothesis
conf.level	confidence level
midp	logical, should the mid-p confidence distribution method be used
nmc	Caculation method when midp=TRUE. If nmc=0 (default) does calculations that are very accurate using uniroot. If nmc>0 does Monte Carlo simulations. The Monte Carlo simulations are not needed for general use.
wmtype	type of modification for the gamma confidence interval, 'max' is the original gamma method that adds max(w) to sum(x*w) for the upper interval, 'mean' adds mean(w), 'minmaxavg' adds mean(c(min(w),max(w)), 'tcz' does a modification of Tiwari, Clegg, and Zou (2006).
mult	a factor to multiply the estimate and confidence intervals by, to give rates per mult
unirootTolFactor	tol factor used in uniroot for calculating when midp=TRUE and nmc=0. Value multiplies by a value close to the quantile of interest in confidence interval, so that if the standardized rates are very small (e.g., 0.00001 before using mult) then the uniroot tol will be unirootTolFactor times that.

Details

Fay and Feuer (1997) developed the gamma method (wmtype='max') for calculating confidence intervals on directly standardized rates. The assumptions is that the k by 1 vector of counts, x, are Poisson with an unknown k by 1 vector of means, theta. There are standardizing weights, w. We are interested in sum(theta*w).

For age-standardization, x is the vector of counts of the event for each of the k age groups. The weights are n.standard/(n.x *sum(n.standard), where n.x[i] is the person-years associated

x[i] and n.standard[i] is person-years fro the standard population associated with the ith age group.

Since the gamma method is conservative, Tiwari, Clegg, and Zou (2006) proposed a modification (wmtype='tcz') and also explored (wmtype='mean').

Ng, Filardo, and Zheng (2008) studied these and other methods (for example, wmtype='minmaxavg') through extensive simulations. They showed that the gamma method (wmtype='max') was the only method that maintained at least nominal coverage in all the simulations. But that method is conservative.

Fay and Kim (2017) proposed the mid-p gamma method. It appears less conservative, while appearing to retain the nominal coverage in almost all simulations. It is calculated by numeric calculations using uniroot.

Value

a list of class htest, containing:

statistic	k=length(x)
parameter	a vector with sample variance of the calibrated weights (so sum(w)=k), and mult (only if mult !=1)
p.value	p-value, set to NA if null.value=NULL
conf.int	confidence interval on true directly standardized rate, sum(theta*w)
estimate	directly standardized rate, sum(x*w)
null.value	null hypothesis value for true DSR
alternative	alternative hypothesis
method	description of method
data.name	desciption of data

Author(s)

Michael P. Fay

References

Fay and Feuer (1997). "Confidence intervals for directly standardized rates: a method based on the gamma distribution." Statistics in Medicine. 16: 791-801.

Fay and Kim (2017). "Confidence intervals for directly standardized rates using mid-p gamma intervals." Biometrical Journal. 59(2): 377-387.

Ng, Filardo, and Zheng (2008). "Confidenc interval estimating procedures for standardized incidence rates." Computational Statistics and Data Analysis 52: 3501-3516.

Tiwari, Clegg, and Zou (2006). "Efficient interval estimation for age-adjusted cancer rates." Statistical Methods in Medical Research. 15: 547-569.

Examples

```
## birth data on Down's syndrome from Michigan, 1950-1964
## see Table II  of Fay and Feuer (1997)
##xfive= counts for mothers who have had 5 or more children
## nfive and ntotal are number of live births
xfive<-c(0,8,63,112,262,295)
nfive<-c(327,30666,123419,149919,104088,34392)
ntotal<-c(319933,931318,786511,488235,237863,61313)
## use mult =10^5 to give rates per 100,000
## gamma method of Fay and Feuer (1997) is default
wspoissonTest(xfive,ntotal/(nfive*sum(ntotal)),mult=10^5)
```

---

wsrTest *Exact Wilcoxon Signed Rank Test*

Description

Calculates the exact Wilcoxon signed rank test (using Pratt's method if there are zero values). Gives exact matching confidence intervals based on repeated calls to `wilcoxsign_test`, and gives associated Hodges-Lehmann estimator of center of the symmetric distribution of the difference.

Usage

```
wsrTest(x, y = NULL, conf.int = TRUE, conf.level = 0.95,
   mu = 0, alternative = c("two.sided", "less", "greater"),
   digits = NULL, tieDigits=8)
```

Arguments

`x`	numeric vector, either the difference (if y=NULL) or the first of the paired responses (so difference is x-y).
`y`	second of paired differences. If NULL assumes x is the vector of paired differences.
`conf.int`	logica, calculate confidence interval on median of differences
`conf.level`	confidence level
`mu`	null median difference
`alternative`	alternative hypothesis
`digits`	number of digits for accuracy of confidence intervals, results are accurate to round(ci,cidigits). If digits=NULL picks about 4 digits if the range of the differences is 0 to 1, with similar accuracy as the range changes (see details).
`tieDigits`	number of digits to round x and y, values closer than that number of digits are treated as tied. This is to avoid rankings based on computer error.

Details

The Wilcoxon signed rank test tests the null hypothesis of whether a set of values (x values, if y=NULL) or differences (x-y, if y!=NULL) are symmetric about mu.

This function calculates the exact Wilcoxon signed rank test using the Pratt method if there are zeros. In other words, rank the differences equal to zero together with the absolute value of the differences, but then permute the signs of only the non-zero ranks. The p-values are calculated using wilcoxsign_test, this function is just a wrapper to get confidence intervals.

When conf.int=TRUE, we get an estimator of the center of the symmetric distribution of the differences based on the shift value where the one-sided p-values are equal (or the middle of the range if there are many values where they are equal). This type of estimator is called a Hodges-Lehmann estimator (see for example, Hodges and Lehmann, 1983). The upper confidence limit when alternative='less' is the smallest shift value that gives a one-sided (alternative='less') p-value that is less than alpha=1-conf.level. Analogously, the lower confidence limit when alternative='greater' is the largest shift value that gives a one-sided (alternative='greater') p-value that is less than alpha. When alternative='two.sided' the confidence interval is the union of the two one-sided intervals each with level 1-alpha/2 (where alpha=1-conf.level). Under the symmetry assumption, the center of a symmetric distribution is its median, pseudo-median, and mean.

Value

An object of class 'htest', list with elements:

estimate	estimator of median difference
p.value	p.value associated with alternative
conf.int	confidence interval
null.value	null median difference
alternative	alternative
method	description of method

Note

The estimator and confidence interval here are different than the ones used in wilcox.test (with paired=TRUE and exact=TRUE).

Author(s)

Michael P. Fay

References

Pratt, JW (1959). Remarks on zeros and ties in the Wilcoxon signed rank procedures. JASA 54(287) 655-667.

Hodges, JL, and Lehmann, EL (1983). Hodges-Lehmann Estimators. In Encyclopedian of Statistics, Volume 3. Editors S. Kotz and NL Johnson. Wiley: New York.

See Also

wilcoxsign_test

Examples

```
wsrTest((-3:8))
```

Index

∗ **datasets**
ama1c1cpg, 4
∗ **htest**
abcnonHtest, 3
anovaOneWay, 5
asht-package, 2
bfTest, 7
cvTest, 9
meldCD, 10
meldtTest, 13
metaNorm, 15
prevSeSp, 17
quantileTest, 20
signTest, 21
simulateSS, 23
tukeyWelsch, 25
var1Test, 28
wmwTest, 29
WprevSeSp, 33
wspoissonTest, 36
wsrTest, 38
∗ **package**
asht-package, 2

abcnon, *4*
abcnonHtest, *2*, 3
ama1c1cpg, 4
anovaOneWay, 5
asht *(*asht-package*)*, 2
asht-package, 2

bfControl, *7*, *13*
bfTest, *2*, 7, *14*
binom.exact, *21, 22*
binomMeld.test, *12*

cvTest, 9

latentTransform, *31*

mcnemarExactDP, *21, 22*
medianTest, *2*, *23*
medianTest *(*quantileTest*)*, 20
meldCD, 10
meldtTest, *12*, 13
metaNorm, 15
methodRuleWMW, *30*
model.frame, *8*, *30*

pairwise.t.test, *26, 27*
pairwise.wilcox.test, *26, 27*
pbf, *8*, *14*
prevSeSp, *2*, 17

quantileTest, *2*, 20

signTest, *2*, *21*, 21
simulateSS, 23

t.test, *8*
tukeyWelsch, 25

var1Test, 28

wilcox.test, *30*, *32*, *39*
wilcox_test, *32*
wilcoxsign_test, *38, 39*
wmwControl, *30*
wmwTest, *2*, 29
WprevSeSp, 33
wspoissonTest, 35
wsrTest, 38