Package ‘exact2x2’

July 22, 2025

Type Package

Title Exact Tests and Confidence Intervals for 2x2 Tables

Version 1.6.9

Date 2024-01-25

Author Michael P. Fay [aut, cre],
Sally A. Hunsberger [ctb],
Martha Nason [ctb],
Erin Gabriel [ctb],
Keith Lumbard [ctb]

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

Depends R (>= 2.10), stats (>= 3.1.1), exactci, ssanv

Description Calculates conditional exact tests (Fisher's exact test, Blaker's exact test, or exact McNemar's test) and unconditional exact tests (including score-based tests on differences in proportions, ratios of proportions, and odds ratios, and Boshcloo's test) with appropriate matching confidence intervals, and provides power and sample size calculations. Gives melded confidence intervals for the binomial case (Fay, et al, 2015, <DOI:10.1111/biom.12231>). Gives boundary-optimized rejection region test (Gabriel, et al, 2018, <DOI:10.1002/sim.7579>), an unconditional exact test for the situation where the controls are all expected to fail. Gives confidence intervals compatible with exact McNemar's or sign tests (Fay and Lumbard, 2021, <DOI:10.1002/sim.8829>). For review of these kinds of exact tests see Fay and Hunsberger (2021, <DOI:10.1214/21-SS131>).

License GPL-3

LazyLoad yes

Suggests testthat, Exact (>= 2.0), ggplot2, grid, gridExtra

NeedsCompilation no

Repository CRAN

Date/Publication 2024-01-25 19:10:02 UTC

Contents

`exact2x2-package` *Exact Tests and Confidence Intervals for 2x2 Tables*

Description

There are 8 main functions in the package. The `exact2x2` function calculates the exact conditional tests with matching confidence intervals as detailed in Fay (2010a <DOI:10.1093/biostatistics/kxp050>,2010b). The functions `ss2x2` and `power2x2` calculate the sample size and power related to the tests of `exact2x2`. The `uncondExact2x2` and `boschloo` functions calculate unconditional exact tests (see Fay and Hunsberger, 2021, <DOI:10.1214/21-SS131>). The `binomMeld.test` function calculates melded confidence intervals for two sample binomial inferences (see Fay, Proschan, and Brittain, 2015 <DOI:10.1111/biom.12231>). Finally, the `borrTest` function calculates the boundary optimized rejection region test that creates unconditional exact tests that have power optimized when group 1 is expected to have 100 percent failure. For example, in vaccine challenge studies where the control group are all expected to get infected (see Gabriel, et al, 2018 <DOI:10.1002/sim.7579>, the letter about that paper by Martin Andres <DOI:10.1002/sim.7630>, and the response <DOI:10.1002/sim.7684>). The `mcnemarExactDP` function give p-values and confidence intervals compatible with exact McNemar's or sign tests (Fay and Lumbard, 2021, <DOI:10.1002/sim.8829>).

Details

Package:	bpcp
Type:	Package
Version:	1.6.9
Date:	2024-01-25
License:	GPL3
LazyLoad:	yes

Author(s)

Michael P. Fay, Sally A. Hunsberger, Martha Nason, Erin Gabriel, Keith Lumbard

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

References

Fay, M. P. (2010a). Confidence intervals that Match Fisher's exact and Blaker's exact tests. Biostatistics, 11: 373-374 (go to doc directory for earlier version or https://www.niaid.nih.gov/about/brb-staff-fay for link to official version).

Fay, M.P. (2010b). Two-sided Exact Tests and Matching Confidence Intervals for Discrete Data. R Journal 2(1):53-58.

Fay, M.P. and Hunsberger, S.A. (2021). Practical Valid Inferences for the Two-Sample Binomial Problem. Statistics Surveys 15:72-110.

Fay, MP, Proschan, MA, and Brittain, E (2015). Combining One Sample Confidence Procedures for Inference in the Two Sample Case. Biometrics. 71: 146-156.

Gabriel, EE, Nason, M, Fay, MP, and Follmann, DA. (2018). A boundary-optimized rejection region test for the two-sample binomial problem. Statistics in Medicine. 37(7): 1047-1058 (DOI: 10.1002/sim.7579).

Gabriel, EE, Nason, M, Fay, MP, and Follmann, DA. (2018). Reply to letter from Martin Andres. Statistics in Medicine 37(14): 2303-2306.

Martin Andres, Antonio. (2018). Letter to the editor about Gabriel et al. Statistics in Medicine 37(14) 2301-2302.

`binomMeld.test` *Melded Binomial Confidence Intervals and Tests*

Description

Creates tests to compare two binomials, giving confidence intervals for either the difference in proportions, the rate ratio, or the odds ratio. The 95 percent confidence intervals have been shown to guarantee nominal coverage by extensive numerical calculations. It has been theoretically proven that the p-values from the one-sided tests on the null hypothesis of equality match Fisher's exact p-values.

Usage

```
binomMeld.test(x1, n1, x2, n2, nullparm = NULL,
    parmtype = c("difference", "oddsratio", "ratio"),
    conf.level = 0.95, conf.int=TRUE,
    alternative = c("two.sided", "less", "greater"),
    midp=FALSE, nmc=0, eps=10^-8)
```

Arguments

`x1`	number of events in group 1
`n1`	sample size in group 1
`x2`	number of events in group 2
`n2`	sample size in group 2
`nullparm`	value of the parameter of interest at null, default of NULL gives 0 for parmtype='difference' and 1 for parmtype='ratio' or 'oddsratio'
`parmtype`	type of parameter of interest, one of "difference", "ratio" or "oddsratio" (see details)
`conf.level`	confidence level
`conf.int`	logical, calculate confidence intervals?
`alternative`	alternative hypothesis, one of "two.sided", "less", or "greater" (see details)
`midp`	logical, do mid-p version of p-value and confidence intervals?
`nmc`	integer, number of Monte Carlo replications for p-value and CI calculations, 0 (default) means calculate by numeric integration instead
`eps`	small number used to adjust numeric integration (see note)

Details

Assume X1~ Binomial(n1,p1) and X2~Binomial(n2,p2). We want to test hypotheses on a function of p1 and p2. The functions are given by parmtype: difference tests p2-p1, ratio tests p2/p1, and odds ratio tests p2(1-p1)/(p1(1-p2)). Let g(p1,p2) be one of the three functions. So when alternative is "less" we test H0: g(p1,p2) >= nullparm vs. H1: g(p1,p2)<nullparm.

For details when `midp=FALSE` see Fay, Proschan, and Brittain (2015).

When `midp=TRUE`, the method performs the mid-p version on the p-value and the associated confidence intervals. This means that we replace the confidence distribution random variables in the p-value and CI calculations with a random variable that is a mixture of the lower and upper CD random variables. For example, if W1L and W1U are the lower and upper confidence distribution random variables for group 1, then we replace those values in all calculations with W1midp = U1*W1L + (1-U1)*W1U, where U1 is a Bernoulli with parameter 0.5. For a discussion of mid-p p-values and the associated confidence intervals in a closely related context, see the vignette on mid p-values or Fay and Brittain (2016, especially the Appendix).

Value

An object of class 'htest'. A list with elements

`statistic`	proportion of events in group 1
`parameter`	proportion of events in group 2
`p.value`	p-value
`conf.int`	confidence interval
`estimate`	estimate of g(p1,p2) by plugging in sample proportions, i.e., unconditional MLE
`null.value`	value of g(p1,p2) under null

`alternative`	type of alternative hypothesis
`method`	description of test
`data.name`	character explicit description of data

Note

For numeric integration, the integrate function may have problems if nearly all of the integrand values are about 0 within the range of integration. Because of this, we use the eps value to make sure we integrate over ranges in which the integrand is nontrivially greater than 0. We restrict the range then add eps back to the p-value so that if the integrate function works perfectly, then the p-values would be very slightly conservative (for very small eps). There is no need to adjust the eps value. See code for detailed description of how eps is used in the calculation before changing it from the default.

An alternative method of calculation is to use Monte Carlo simulation (option with `nmc>0`). This provides a check of the numeric integration. There is no need to do Monte Carlo simulations for routine use. Please inform the package maintainer if the p-values or confidence intervals are substantially different when `nmc=0` and `nmc=10^7`.

Author(s)

Michael P. Fay

References

Fay, MP, Proschan, MA, and Brittain, E (2015) Combining One Sample Confidence Procedures for Inferences in the Two Sample Case. Biometrics 71: 146-156.

Fay, Michael P., and Erica H. Brittain. (2016). Finite sample pointwise confidence intervals for a survival distribution with right-censored data. Statistics in medicine. 35: 2726-2740.

Examples

```
# Note the p-value for all tests of equality
# (Null Hypthesis: true prop 1=true prop 2)
# are the same, and equal to the
# Fisher's exact (central) p-value
binomMeld.test(3,5,1,8,parmtype="difference")
binomMeld.test(3,5,1,8,parmtype="ratio")
# note that binomMeld.test gives the unconditional MLE
# for the odds ratio, while fisher.test and exact2x2
# gives the conditional MLE for the odds ratio
# (also fisher.test gives the odds ratio defined as
#  the inverse of how it is defined in binomMeld.test)
binomMeld.test(3,5,1,8,parmtype="oddsratio")
exact2x2(matrix(c(1,8-1,3,5-3),2,2),tsmethod="central")
```

`borrTest` *Boundary-Optimized Rejection Region Test*

Description

An unconditional exact test for the two-sample binomial problem when it is expected that theta1 (probability of an event in group 1) will be close to 1. Used for test versus control when all controls are expected to fail.

Usage

```
borrTest(x1, n1, x2, n2, tuningParm = 0.025,
    parmtype = c("ratio", "difference", "oddsratio"),
    nullparm = NULL, alternative = c("less", "greater", "two.sided"),
    conf.int = TRUE, conf.level = 0.975,
    controlUC = ucControl(), controlborr = borrControl(), ...)

borrPvals(n1,n2, tuningParm=0.025,
    parmtype = c("ratio", "difference","oddsratio"),
    nullparm = NULL, alternative = c("less", "greater","two.sided"),
    conf.int = TRUE, conf.level = 0.975,
    controlUC=ucControl(), controlborr=borrControl(),...)

borrOrdering(n1,n2,tuningParm = .025,
            controlborr=borrControl())

powerBorr(n1,n2,p1,p2,alpha=0.025,...)
```

Arguments

`x1`	number of events in group 1
`n1`	sample size in group 1
`x2`	number of events in group 2
`n2`	sample size in group 2
`tuningParm`	tuning parameter, default is 0.025 and designs BORR tests with maximum power for one-sided 0.025 tests
`parmtype`	parameter type, either 'ratio' for theta2/theta1, 'difference' for theta2-theta1, or 'oddsratio' for theta2*(1-theta1)/(theta1*(1-theta2)).
`nullparm`	null parameter value, default=NULL gives parameter value for theta1=theta2 (e.g., 1 for 'ratio' or 0 for 'difference').
`alternative`	alternative hypothesis, BORR tests are designed for alternative='less' (see Note for other alternatives)
`conf.int`	logical, should confidence interval be calculated?
`conf.level`	confidence level, default is 0.975 (see note)

`controlUC`	a list of control parameters to define algorithms in the call to `uncondExact2x2`, see `ucControl`
`controlborr`	a list of control parameters to define algorithms, see `borrControl`
`p1`	probability of an event in group 1
`p2`	probability of an event in group 2
`alpha`	alpha-level for rejecting, reject when p-value $latex$ alpha
`...`	extra arguments passed (only used for `powerBorr`, passes arguments to the `borrPvals` function)

Details

The boundary-optimized rejection region test is designed to test the one-sided alternative that theta2 < theta1, where X1 is binomial(n1,theta1), and X2 is binomial(n2,theta2). The test is designed to be optimal when theta1 is very close to 1. For example, in a vaccine malaria challenge study where we expect all n1 individuals that got the control vaccine to have the event (get malaria when challenged with malaria). For details see Gabriel et al (2018).

The function `borrTest` tests the results of one study, and returns an `htest` object. The function `borrPvals` calculates the p-values for every possible result of a study. The function `borrOrdering` orders every possible result of the study. See `borrOrderingInternal` for calculation details. The function `powerBorr` calculates the power where p-values are calculated by `borrPvals` and rejection is when

$$latex$$

alpha.

Value

The function `borrPvals` returns a (n1+1) by (n2+1) matrix of p-values for all possible x1 and x2 values. The function `borrOrdering` returns a matrix with the rank of all possible x1 and x2 values. The function `borrTest` returns a list of class `htest` with elements:

`statistic`	proportion in sample 1
`parameter`	proportion in sample 2
`p.value`	p-value from test
`conf.int`	confidence interval on parameter given by parmtype
`estimate`	MLE estimate of parameter given by parmtype
`null.value`	null hypothesis value of parameter given by parmtype
`alternative`	alternative hypothesis
`method`	description of test
`data.name`	description of data

Note

The tests are designed to have good power for the one-sided test that H0: theta2 $\geq$ theta1, with alternative H1: theta2 < theta1 at significance level equal to `tuningParm`. Since the default `tuningParm` is 0.025, the default confidence level is 0.975 so that the confidence intervals will be compatible with the test where the one-sided p-values reject at level 0.025.

Sometimes you may want two-sided confidence intervals on the parameter of interest. If you ask for a two-sided alternative, then the confidence interval and the resulting p-value will be two-sided as well. The default is a 'central' interval, so the two-sided p-value should be twice the minimum of the one-sided p-values. Further, with a `conf.level` of 0.95 for the two-sided alternative, the error on each side will be bounded by 0.025.

Author(s)

Martha Nason, Erin Gabriel, Michael P. Fay

References

Gabriel, EE, Nason, M, Fay, MP, and Follmann, DA. (2018). A boundary-optimized rejection region test for the two-sample binomial problem. Statistics in Medicine. 37(7): 1047-1058 (DOI: 10.1002/sim.7579).

Gabriel, EE, Nason, M, Fay, MP, and Follmann, DA. (2018). Reply to letter from Martin Andres. Statistics in Medicine 37(14): 2303-2306.

Martin Andres, Antonio. (2018). Letter to the editor about Gabriel et al. Statistics in Medicine 37(14) 2301-2302.

Examples

```
## Not run: borrTest(4,4,1,4)
# Note Figure 2 in Gabriel et al is incorrect. The correct value
# is in the response letter, and given by
borrOrdering(4,4,tuningParm=0.025)$rankMat
```

`boschloo` *Boschloo's test for 2x2 Tables*

Description

Boschloo's test is an exact unconditional test for 2x2 tables based on ordering the sample space by Fisher's exact p-values. This function generalizes that test in several ways (see details).

Usage

```
boschloo(x1, n1, x2, n2, alternative = c("two.sided", "less", "greater"),
    or = NULL, conf.int = FALSE, conf.level = 0.95, midp = FALSE,
    tsmethod = c("central", "minlike"), control=ucControl())
```

Arguments

`x1`	number of events in group 1
`n1`	sample size in group 1
`x2`	number of events in group 2
`n2`	sample size in group 2
`alternative`	alternative hypothesis, one of "two.sided", "less", or "greater", default is "two.sided" (see details)
`or`	odds ratio under the null hypothesis
`conf.int`	logical, calculate confidence interval?
`conf.level`	confidence level
`midp`	logical. Use mid-p-value method?
`tsmethod`	two-sided method, either "central" or "minlike" (see details)
`control`	list of algorithm parameters, see `ucControl`

Details

The traditional Boschloo (1970) test is to use Fisher's exact p-values (under the null that p1=p2) to order the sample space and to use that ordering to perform an unconditional exact test. Here we generalize this to test for different null hypothesis values (other that odds ratios of 1).

For the two-sided alternatives, the traditional method uses tsmethod='minlike' (for example, in the Exact R package) but our default is tsmethod='central'. The one-sided tests use ordering by the appropriate p-value (or 1 minus the p-value for alternative='greater' so that the ordering function follows our convention for user supplied ordering functions, see method='user' option in `uncondExact2x2`).

The option `midp` orders the sample space by the mid-p value associated with Fisher's exact test, and additionally gives mid-p values. This means that unlike the `midp=FALSE` case, when `midp=TRUE` the test is not exact (i.e., guaranteed to bound the type I error rate at the nominal level), but has type I error rates that are on average (over the possible null parameter values) closer to the nominal level.

If you want to order by the mid-p values from Fisher's exact test but get an exact test, use the `method="FisherAdj"` with `midp=FALSE` in `uncondExact2x2`.

The `boschloo` function only gives confidence intervals for the odds ratio, for getting confidence intervals closely related to Boschloo p-values (but not exactly matching Boschloo p-values) for the difference or ratio, use `uncondExact2x2` with `method="FisherAdj"`.

Value

a list of class 'htest' with elements:

`statistic`	proportion in sample 1
`parameter`	proportion in sample 2
`p.value`	p-value from test
`conf.int`	confidence interval on odds ratio
`estimate`	odds ratio estimate

`null.value`	null hypothesis value of odds ratio
`alternative`	alternative hypothesis
`method`	description of test
`data.name`	description of data

References

Boschloo, R. D. "Raised conditional level of significance for the 2x2-table when testing the equality of two probabilities." Statistica Neerlandica 24.1 (1970): 1-9.

See Also

`exact.test` in package **Exact** for Boschloo test p-value computation. Also see `method"FisherAdj"` in `uncondExact2x2` for a closely related test.

Examples

```
# defaults to the central two-sided version
boschloo(1,5,6,7)
boschloo(1,5,6,7,alternative="greater")
## traditional two-sided Boschloo test (not central!)
boschloo(1,5,6,7, tsmethod="minlike")
```

`exact2x2` *Exact Conditional Tests for 2 by 2 Tables of Count Data*

Description

Performs exact conditional tests for two by two tables. For independent binary responses, performs either Fisher's exact test or Blaker's exact test for testing hypotheses about the odds ratio. The commands follow the style of `fisher.test`, the difference is that for two-sided tests there are three methods for calculating the exact test, and for each of the three methods its matching confidence interval is returned (see details). For paired binary data resulting in a two by two table, performs an exact McNemar's test.

Usage

```
exact2x2(x, y = NULL, or = 1, alternative = "two.sided",
    tsmethod = NULL, conf.int = TRUE, conf.level = 0.95,
    tol = 0.00001, conditional = TRUE, paired=FALSE,
    plot=FALSE, midp=FALSE)
fisher.exact(x, y = NULL, or = 1, alternative = "two.sided",
    tsmethod = "minlike", conf.int = TRUE, conf.level = 0.95,
    tol = 0.00001, midp=FALSE)
blaker.exact(x, y = NULL, or = 1, alternative = "two.sided",
    conf.int = TRUE, conf.level = 0.95, tol = 0.00001)
mcnemar.exact(x,y=NULL, conf.level=.95)
```

Arguments

`x`	either a two-dimensional contingency table in matrix form, or a factor object.
`y`	a factor object; ignored if `x` is a matrix.
`or`	the hypothesized odds ratio. Must be a single numeric.
`alternative`	indicates the alternative hypothesis and must be one of `"two.sided"`, `"greater"` or `"less"`. if "two.sided" uses method defined by tsmethod.
`tsmethod`	one of "minlike","central", or "blaker". NULL defaults to "minlike" when paired=FALSE and "central" when paired=TRUE or midp=TRUE. Defines type of two-sided method (see details). Ignored if alternative="less" or "greater".
`conf.int`	logical indicating if a confidence interval should be computed.
`conf.level`	confidence level for the returned confidence interval. Only used if `conf.int = TRUE`.
`tol`	tolerance for confidence interval estimation.
`conditional`	TRUE. Unconditional exact tests should use `uncondExact2x2`.
`paired`	logical. TRUE gives exact McNemar's test, FALSE are all other tests
`midp`	logical. TRUE gives mid p-values and mid-p CIs. Not supported for tsmethod='minlike' or 'blaker'
`plot`	logical. TRUE gives basic plot of point null odds ratios by p-values, for greater plot control use `exact2x2Plot`. Not supported for midp=TRUE.

Details

The motivation for this package is to match the different two-sided conditional exact tests for 2x2 tables with the appropriate confidence intervals.

There are three ways to calculate the two-sided conditional exact tests, motivated by three different ways to define the p-value. The usual two-sided Fisher's exact test defines the p-value as the sum of probability of tables with smaller likelihood than the observed table (`tsmethod`="minlike"). The central Fisher's exact test defines the p-value as twice the one-sided p-values (but with a maximum p-value of 1). Blaker's (2000) exact test defines the p-value as the sum of the tail probibility in the observed tail plus the largest tail probability in the opposite tail that is not greater than the observed tail probability.

In `fisher.test` the p-value uses the two-sample method associated with `tsmethod`="minlike", but the confidence interval method associated with `tsmethod`="central". The probability that the lower central confidence limit is less than the true odds ratio is bounded by `1-(1-conf.level)/2` for the central intervals, but not for the other two two-sided methods. The confidence intervals in for `exact2x2` match the test associated with alternative. In other words, the confidence interval is the smallest interval that contains the confidence set that is the inversion of the associated test (see Fay, 2010). The functions `fisher.exact` and `blaker.exact` are just wrappers for certain options in `exact2x2`.

If `x` is a matrix, it is taken as a two-dimensional contingency table, and hence its entries should be nonnegative integers. Otherwise, both `x` and `y` must be vectors of the same length. Incomplete cases are removed, the vectors are coerced into factor objects, and the contingency table is computed from these.

P-values are obtained directly using the (central or non-central) hypergeometric distribution.

The null of conditional independence is equivalent to the hypothesis that the odds ratio equals one. 'Exact' inference can be based on observing that in general, given all marginal totals fixed, the first element of the contingency table has a non-central hypergeometric distribution with non-centrality parameter given by the odds ratio (Fisher, 1935). The alternative for a one-sided test is based on the odds ratio, so `alternative = "greater"` is a test of the odds ratio being bigger than `or`.

When paired=TRUE, this denotes there is some pairing of the data. For example, instead of Group A and Group B, we may have pretest and posttest binary responses. The proper two-sided test for such a setup is McNemar's Test, which only uses the off-diagonal elements of the 2x2 table, and tests that both are equal or not. The exact version is based on the binomial distribution on one of the off-diagonal values conditioned on the total of both off-diagonal values. We use `binom.exact` from the `exactci` package, and convert the p estimates and confidence intervals (see note) to odds ratios (see Breslow and Day, 1980, p. 165). The function `mcnemar.exact` is just a wrapper to call `exact2x2` with `paired=TRUE, alternative="two.sided",tsmethod="central"`. One-sided exact McNemar-type tests may be calculated using the `exact2x2` function with `paired=TRUE`. For details of McNemar-type tests see Fay (2010, R Journal).

The mid p-value is an adjusted p-value to account for discreteness. The mid-p adjustment is not guaranteed to give type I error rates that are less than or equal to nominal levels, but gives p-values that lead to the probability of rejection that is sometimes less than the nominal level and sometimes greater than the nominal level. This adjustment is sometimes used because exact p-values for discrete data cannot give actual type I error rates equal to the nominal value unless randomization is done (and that is not typically done because two researchers doing the same method could get different answers). Essentially, exact p-values lead to the probability of rejecting being less than the nominal level for most parameter values in the null hypothesis in order to make sure that it is not greater than the nominal level for ANY parameter values in the null hypothesis. The mid p-value was studied by Lancaster (1961), and for the 2x2 case by Hirji et al (1991).

Value

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

`p.value`	the p-value of the test
`conf.int`	a confidence interval for the odds ratio
`estimate`	an estimate of the odds ratio. Note that the *conditional* Maximum Likelihood Estimate (MLE) rather than the unconditional MLE (the sample odds ratio) is used.
`null.value`	the odds ratio under the null, `or`.
`alternative`	a character string describing the alternative hypothesis
`method`	a character string, changes depending on alternative and tsmethod
`data.name`	a character string giving the names of the data

Note

The default exact confidence intervals for the odds ratio when paired=TRUE (those matching the exact McNemar's test) are transformations of the Clopper-Pearson exact confidence intervals for a single binomial parameter which are central intervals. See note for `binom.exact` for discussion of exact binomial confidence intervals.

Author(s)

Michael Fay

References

Blaker, H. (2000) Confidence curves and improved exact confidence intervals for discrete distributions. Canadian Journal of Statistics 28: 783-798.

Breslow, NE and Day NE (1980). Staistical Methods in Cancer Research: Vol 1-The analysis of Case-Control Studies. IARC Scientific Publications. IARC, Lyon.

Fay, M. P. (2010). Confidence intervals that Match Fisher's exact and Blaker's exact tests. Biostatistics, 11: 373-374 (go to doc directory for earlier version or https://www.niaid.nih.gov/about/brb-staff-fay for link to official version).

Fay M.P. (2010). Two-sided Exact Tests and Matching Confidence Intervals for Discrete Data. R Journal 2(1):53-58.

Fisher, R.A. (1935) The logic of inductive inference. Journal of the Royal Statistical Society Series A 98:39-54.

Hirji, K.F., Tan, S-J, and Elashoff, R.M. (1991). A quasi-exact test for comparing two binomial proportions. Statistics in Medicine 10: 1137-1153.

Lancaster, H.O. (1961). Significance tests in discrete distributions. JASA 56: 223-234.

See Also

`fisher.test` or `mcnemar.test`

Examples

```
## In example 1, notice how fisher.test rejects the null at the 5 percent level,
## but the 95 percent confidence interval on the odds ratio contains 1
## The intervals do not match the p-value.
## In fisher.exact you get p-values and the matching confidence intervals
example1<-matrix(c(6,12,12,5),2,2,dimnames=list(c("Group A","Group B"),c("Event","No Event")))
example1
fisher.test(example1)
fisher.exact(example1,tsmethod="minlike")
fisher.exact(example1,tsmethod="central")
blaker.exact(example1)
## In example 2, this same thing happens, for
## tsmethod="minlike"... this cannot be avoided because
## of the holes in the confidence set.
##
example2<-matrix(c(7,255,30,464),2,2,dimnames=list(c("Group A","Group B"),c("Event","No Event")))
example2
fisher.test(example2)
exact2x2(example2,tsmethod="minlike")
## you can never get a test-CI inconsistency when tsmethod="central"
exact2x2(example2,tsmethod="central")
```

---

exact2x2Plot *Plot p-value function for one 2 by 2 table.*

Description

Plots two-sided p-values as a function of odds ratios. Can plot three types of p-values: the two-sided Fisher's exact, the central Fisher's exact (i.e., twice the one-sided Fisher's exact), and Blaker's exact.

Usage

```
exact2x2Plot(x, y=NULL, OR = NULL, ndiv = 1000, tsmethod=NULL,
    method = NULL, paired=FALSE, orRange = NULL, dolog = TRUE,
    dolines = FALSE, dopoints = TRUE, doci=TRUE,
    alternative=c("two.sided","less","greater"),
    conf.level=.95, alphaline=TRUE, newplot = TRUE, ...)
```

Arguments

`x`	matrix representing the 2 by 2 table
`y`	a factor object; ignored if `x` is a matrix.
`OR`	odds ratio values for plot, if NULL divides orRange into ndiv pieces
`ndiv`	number of pieces to divide up odds ratio range
`tsmethod`	either "minlike","blaker" or "central"
`method`	same as tsmethod, kept for backward compatability
`paired`	logical, do paired analysis giving McNemar's test p-values
`orRange`	range for calculating odds ratios
`dolog`	logical,plot odds ratios on log scale?
`dolines`	logical, add lines to a plot?
`dopoints`	logical, add points to a plot?
`doci`	logical, add vertical lines at confidence interval?
`alternative`	one of "two.sided","less","greater", type of alternative for p-values
`conf.level`	when doci=TRUE, level for confidence interval to be plotted
`alphaline`	logical, if doci=TRUE should a line be drawn at the significance level?
`newplot`	logical,start a new plot?
`...`	values passed to plot, points, or lines statement

See Also

`exact2x2`

Examples

```
example1<-matrix(c(6,12,12,5),2,2,dimnames=list(c("Group A","Group B"),c("Event","No Event")))
example1
exact2x2Plot(example1)
## add lines from central Fisher's exact
exact2x2Plot(example1,method="central",dolines=TRUE,newplot=FALSE,col="red")
```

`mcnemarExactDP` *Exact McNemar (Paired Binary) Test with Difference in Proportions*

Description

Gives a valid (i.e., exact) test of paired binary responses, with compatible confidence intervals on the difference in proportions.

Usage

```
mcnemarExactDP(x, m, n, nullparm = 0, alternative = c("two.sided", "less", "greater"),
    conf.level = 0.95, nmc = 0)
```

Arguments

`m`	number of pairs with mismatched responses
`x`	number of pairs with response of 1 for treatment and 0 for control
`n`	total number of pairs
`nullparm`	null parameter value for the difference in proportions: proportion with events on treatment minus proportion with events on control
`alternative`	alternative hypothesis, must be one of "two.sided", "greater" or "less"
`conf.level`	confidence level for the returned confidence interval
`nmc`	number of Monte Carlo replications, nmc=0 (default) uses numeric integration instead

Details

For paired binary responses, a simple test is McNemars test, which conditions on the number of discordant pairs. The `mcnemar.exact` function gives results in terms of odds ratios. This function gives results in terms of the difference in proportions. The p-values will be identical between the two functions, but the estimates and confidence intervals will be different.

For this function, we use the melding idea (Fay, et al, 2015), to create compatable confidence intervals with exact versions of McNemars test. For details see Fay and Lumbard (2021). See Fagerland, et al (2013) for other parameters and methods related to paired binary responses. The advantage of this version is that it is exact, and faster than the unconditional exact methods (which may be more powerful).

Value

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

`p.value`	the p-value of the test
`conf.int`	a confidence interval for the difference in proportions
`estimate`	sample proportions and their difference
`null.value`	difference in proportions under the null
`alternative`	a character string describing the alternative hypothesis
`method`	a character string describing the test
`data.name`	a character string giving the names of the data

Author(s)

Michael P. Fay, Keith Lumbard

References

Fay, MP, Proschan, MA, and Brittain, E (2015). Combining one-sample confidence procedures for inference in the two-sample case. Biometrics,71(1),146-156.

Fay MP, and Lumbard, K (2021). Confidence Intervals for Difference in Proportions for Matched Pairs Compatible with Exact McNemars or Sign Tests. Statistics in Medicine, 40(5): 1147-1159.

Fagerland, Lydersen and Laake (2013), Recommended tests and confidence intervals for paired binomial proportions. Statitics in Medicine, 33:2850-2875.

See Also

See `mcnemar.exact` or `exact2x2` with paired=TRUE for confidence intervals on the odds ratio.

Examples

```
# For test on contingency table of the pairs
# From Bentur, et al (2009) Pediatric Pulmonology 44:845-850.
# see also Table II of Fagerland, Lydersen and Laake
# (2013, Stat in Med, 33: 2850-2875)
#
#                 After SCT
#                 AHR       No AHR
#              -----------------
# Before SCT  |
#         AHR |    1         1
#      No AHR |    7        12
#              -----------------

ahr<-matrix(c(1,7,1,12),2,2,
            dimnames=list(paste("Before SCT,",c("AHR","No AHR")),
                          paste("After SCT,",c("AHR","No AHR"))))
mcnemarExactDP(n=sum(ahr),m=ahr[1,2]+ahr[2,1], x=ahr[1,2])
# compare to mcnemar.exact
```

```
# same p-value, but mcnemar.exact gives conf int on odds ratio
mcnemar.exact(ahr)
```

`plotT` *Plot or Print ordering function for unconditional exact test*

Description

The function `orderMat` prints the values for the ordering function for all possible values of X1 and X2 in matrix form.

The function `plotT` plots the ranking of the ordering function on an n1+1 by n2+1 grid, where each square represents a possible values for (x1,x2). The default colors are from dark blue (highest) to light blue to white (middle) to light red to dard red (lowest), with black=NA.

Usage

```
plotT(x, ...)

## S3 method for class 'function'
plotT(x, n1, n2, delta0 = 1, main = "",...)


## S3 method for class 'numeric'
plotT(x, n1, n2, delta0 = 1, main = "",...)


orderMat(x, ...)

## S3 method for class 'function'
orderMat(x,n1,n2,delta0,graphStyle=FALSE,...)

## S3 method for class 'numeric'
orderMat(x,n1,n2,delta0,graphStyle=FALSE,...)
```

Arguments

`x`	object, either a Tstat function, or a vector of all (n1+1)*(n2+1) possible values of the function (see details).
`n1`	sample size in group 1
`n2`	sample size in group 2
`delta0`	null value of parameter (if needed for Tstat function)
`main`	plot title
`graphStyle`	logical, order rows with lowest x1 value on the bottom?
`...`	arguments to be passed to the Tstat function

Details

If `x` is all the values of the Tstat function, then the values should be ordered by cycling through the x1 values (0 to n1) for each x2 value. Specifically, it should be the result of `Tstat(X1,n1,X2,n2,delta0)` where X1=rep(0:n1,n2+1) and X2=rep(0:n2,each=n1+1).

Examples

```
parorig<- par(no.readonly=TRUE)
par(mfrow=c(2,2),mar=c(1,3,3,1))
TT1<-pickTstat(method="score", parmtype="ratio", tsmethod="central", alternative="two.sided")
round(orderMat(TT1,8,8,1,graphStyle=TRUE),2)
TT2<-pickTstat(method="simple", parmtype="ratio", tsmethod="central", alternative="two.sided")
TT3<-pickTstat(method="simple", parmtype="difference", tsmethod="central", alternative="two.sided")
plotT(TT2, 8,8, 1, main="Ratio, Simple")
plotT(TT3, 8,8, 0, main="Difference, Simple")
plotT(TT1, 8,8, 1, main="Ratio, Score (delta0=1)")
TF<-pickTstat(method="FisherAdj", parmtype="ratio", tsmethod="central", alternative="two.sided")
plotT(TF,8,8,1, main="FisherAdj")
par(parorig)
```

`power2x2` *Calculate exact power or sample size for conditional tests for two independent binomials.*

Description

Power is calculated by `power2x2` which calls `exact2x2` function repeatedly. Default (strict=FALSE) does not count rejections in the wrong direction.

Sample size is calculated by `ss2x2` which calls `power2x2` repeatedly finding the lowest sample size that has at least the nominal power, using the `uniroot.integer` function from the `ssanv` package.

Usage

```
power2x2(p0,p1,n0,n1=NULL,sig.level=0.05,
 alternative=c("two.sided","one.sided"),paired=FALSE,
 strict=FALSE,tsmethod=NULL,nullOddsRatio=1,
 errbound=10^-6,approx=FALSE)

ss2x2(p0,p1,power=.80,n1.over.n0=1,sig.level=0.05,
 alternative=c("two.sided","one.sided"),paired=FALSE,
 strict=FALSE,tsmethod=NULL,nullOddsRatio=1,
 errbound=10^-6,print.steps=FALSE, approx=FALSE)
```

Arguments

`p0`	true event rate in control group
`p1`	true event rate in treatment group
`n0`	number of observations in control group
`n1`	number of observations in treatment group (if NULL n1=n0)
`sig.level`	significance level (Type I error probability)
`power`	minimum power for sample size calculation
`n1.over.n0`	ratio of n1 over n0, allows for non-equal sample size allocation
`alternative`	character, either "two.sided" or "one.sided", one sided tests the proper direction according to p0 and p1
`strict`	use strict interpretation of two-sided test, if TRUE counts rejections in wrong direction
`tsmethod`	two.sided method, ignored if strict=FALSE, or alternative equals 'less' or 'greater'. see `exact2x2` for details.
`nullOddsRatio`	null odds ratio value for tests
`paired`	must be FALSE, for TRUE instead use `powerPaired2x2`
`print.steps`	logical, print steps for calculation of sample size?
`errbound`	bound on error of calculation
`approx`	give sample size or power using normal approximation only

Details

Assuming X0 ~ Binomial(n0,p0) and X1 ~ Binomial(n1,p1), calculates the power by repeatedly calling exact2x2 and summing probability of rejection. For speed, the function does not calculate the very unlikely values of X0 and X1 unless errbound=0. Power is exact, but may underestimate by at most errbound.

When strict=FALSE we do not count rejections in the wrong direction. This means that we must know the direction of the rejection, so two.sided tests are calculated as one.sided tests (in the correct direction) with level equal to sig.level/2. This is like using the tsmethod='central'.

When `approx`=TRUE for `power2x2` use a continuity corrected normal approximation (Fleiss, 1981, p. 44). For `ss2x2` the calculations may be slow, so use `print.steps`=TRUE to see progress.

Value

Both `power2x2` and `ss2x2` return an object of class 'power.htest'. A list with elements

`power`	power to reject
`n0`	sample size in control group
`n1`	sample size in treatment group
`p0`	true event rate in control group
`p1`	true event rate in treatment group
`sig.level`	Significance level (Type I error probability)
`alternative`	alternative hypothesis
`note`	note about error bound
`method`	description

Warning

There may be convergence issues using strict=FALSE with tsmethod="minlike" or "blaker" since the power is not guaranteed to be increasing in the sample size.

Note

The calculations in ss2x2 can be slow when p0 is close to p1 and/or the power is large. If p0 and p1 are close with large power, it may be safer to first calculate ss2x2 with approx=TRUE to see what the starting value will be close to. If the starting sample sizes are large (>100), it may take a while.

Note when strict=FALSE (default), the two.sided results at the 0.05 level for Fisher's exact test are like the one.sided Fisher's exact test at the 0.025 level.

Author(s)

Michael P. Fay

References

Fleiss. JL (1981) Statistical Methods for Rates and Proportions (second edition). Wiley.

See Also

See `ss.nonadh` function (refinement="Fisher.exact") from the `ssanv` package for calculation that accounts for nonadherence in proportion of subjects. That function calls `fisher.test`. For power for McNemar-like test see `powerPaired2x2`

Examples

```
power2x2(.2,.8,12,15)
# calculate sample size with 2:1 allocation to groups
ss2x2(.2,.8,n1.over.n0=2,power=.8,approx=TRUE)
ss2x2(.2,.8,n1.over.n0=2,power=.8,print.steps=TRUE)
```

`powerPaired2x2` *Power for exact McNemar's test*

Description

Calculate the power for the exact McNemar's test (i.e., `exact2x2` with `paired=TRUE`) given the number of pairs and the probability of a positive response only in the test individual in the pair (pb), and the probability of a positive response only in the control individual in the pair (pc).

Usage

```
powerPaired2x2(pb, pc, npairs, sig.level = 0.05,
   alternative = c("two.sided", "one.sided"),
   strict = FALSE, nullOddsRatio = 1, errbound = 10^-6, ...)
```

Arguments

`pb`	probability of a (0,1) response for a pair, meaning negative response in the control individual and a positive response in the test individual
`pc`	probability of a (1,0) response for a pair, meaning positive response in the control individual and a negative response in the test individual
`npairs`	the number of pairs
`sig.level`	significance level (also called alpha-level)
`alternative`	either 'one.sided' or 'two.sided' (see `tsmethod` for two-sided method)
`strict`	use strict interpretation in two-sided case (i.e., TRUE allows rejections in the 'wrong' direction)
`nullOddsRatio`	null odds ratio, internally passed to `or` argument of `exact2x2` with `paired=TRUE`
`errbound`	error bound, `errbound=0` does exact calculation, when `errbound>0` then speed up calculations by not calculating outcomes at either extreme with tail probabilities less than errbound/2 which may underestimate power by at most errbound.
`...`	arguments passed to `exact2x2` (except these arguments cannot be passed this way: `or`, `alternative`, `conf.int`, `paired`, `plot`)

Details

When `alternative='one.sided'` then the test automatically picks the side that is most powerful. At this point there is no ssPaired2x2 function.

Value

An object of class 'power.htest' with elements:

`power`	power
`npairs`	number of pairs
`pb`	probability of a (control,test)=(0,1) response for a pair
`pc`	probability of a (control,test)=(1,0) response for a pair
`sig.level`	significance level or alpha-level
`alternative`	either one-sided or two-sided
`nullOddsRatio`	null odds ratio (or boundary between null and alternative for one-sided tests)
`note`	notes about calculation (e.g., errbound value)
`method`	description of method

Examples

```
powerPaired2x2(.5,.3,npairs=20)
```

`uncondExact2x2` *Unconditional exact tests for 2x2 tables*

Description

The `uncondExact2x2` function tests 2x2 tables assuming two independent binomial responses. Unlike the conditional exact tests which condition on both margins of the 2x2 table (see `exact2x2`), these unconditional tests only condition on one margin of the 2x2 table (i.e., condition on the sample sizes of the binomial responses). This makes the calculations difficult because now there is a nuisance parameter and calculations must be done over nearly the entire nuisance parameter space.

Usage

```
uncondExact2x2(x1, n1, x2, n2,
    parmtype = c("difference", "ratio", "oddsratio"), nullparm = NULL,
    alternative = c("two.sided","less", "greater"),
    conf.int = FALSE, conf.level = 0.95,
  method = c("FisherAdj", "simple", "score","wald-pooled", "wald-unpooled",  "user",
     "user-fixed"),
    tsmethod = c("central","square"), midp = FALSE,
    gamma = 0, EplusM=FALSE, tiebreak=FALSE,
    plotprobs = FALSE, control=ucControl(), Tfunc=NULL,...)

uncondExact2x2Pvals(n1, n2, ...)
```

Arguments

`x1`	number of events in group 1
`n1`	sample size in group 1
`x2`	number of events in group 2
`n2`	sample size in group 2
`parmtype`	type of parameter of interest, one of "difference", "ratio" or "oddsratio" (see details)
`nullparm`	value of the parameter of interest at null hypothesis, NULL defaults to 0 for parmtype='difference' and 1 for parmtype='ratio' or 'oddsratio'
`alternative`	alternative hypothesis, one of "two.sided", "less", or "greater", default is "two.sided" (see details)
`conf.int`	logical, calculate confidence interval?
`conf.level`	confidence level
`method`	method type, one of "FisherAdj" (default), "simple", "simpleTB", "wald-pooled", "wald-unpooled", "score", "user", or "user-fixed" (see details)
`tsmethod`	two-sided method, either "central" or "square" (see details)
`midp`	logical. Use mid-p-value method?

`gamma`	Beger-Boos adjustment parameter. 0 means no adjustment. (see details).
`EplusM`	logical, do the E+M adjustment? (see details)
`tiebreak`	logical, do tiebreak adjustment? (see details)
`plotprobs`	logical, plot probabilities?
`control`	list of algorithm parameters, see `ucControl`
`Tfunc`	test statistic function for ordering the sample space when method='user', ignored otherwise (see details)
`...`	extra arguments passed to Tfunc (for uncondExact2x2), or passed to uncondExact2x2 (for uncondExact2x2Pvals)

Details

The `uncondExact2x2` function gives unconditional exact tests and confidence intervals for two independent binomial observations. The `uncondExact2x2Pvals` function repeatedly calls `uncondExact2x2` to get the p-values for the entire sample space.

Let X1 be binomial(n1,theta1) and X2 be binomial(n2,theta2). The parmtype determines the parameter of interest: ‘difference’ is theta2 - theta1, ’ratio’ is theta2/theta1, and ‘oddsratio’ is (theta2*(1-theta1))/(theta1*(1-theta2)).

The options `method`, `parmtype`, `tsmethod`, `alternative`, `EplusM`, and `tiebreak` define some built-in test statistic function, Tstat, that is used to order the sample space, using `pickTstat` and `calcTall`. The first 5 arguments of Tstat must be `Tstat(X1,N1,X2,N2, delta0)`, where X1 and X2 must allow vectors, and delta0 is the null parameter value (but delta0 does not need to be used in the ordering). Ordering when `parmtype="ratio"` or `parmtype="oddsratio"` is only used when there is information about the parameter. So the ordering function value is not used for ordering when x1=0 and x2=0 for `parmtype="ratio"`, and it is not used when (x1=0 and x2=0) or (x1=n1 and x2=n2) for `parmtype="oddsratio"`.

We describe the ordering functions first for the basic case, the case when `tsmethod="central"` or `alternative!="two.sided"`, `EplusM=FALSE`, and `tiebreak=FALSE`. In this basic case the ordering function, Tstat, is determined by `method` and `parmtype`:

- method=’simple’ - Tstat essentially replaces theta1 with x1/n1 and theta2 with x2/n2 in the parameter definition. If parmtype=‘difference’ then `Tstat(X1,N1,X2,N2,delta0)` returns `X2/N2-X1/N1-delta0`. If parmtype=’ratio’ then the Tstat function returns `log(X2/N2) - log(X1/N1) - log(delta0)`. If parmtype=’oddsratio’ we get `log( X2*(N1-X1)/(delta0*X1*(N2-X2)))`.
- method=’wald-pooled’ - Tstat is a Z statistic on the difference using the pooled variance (not allowed if `parmtype!="difference"`)
- method=’wald-unpooled’ - Tstat is a Z statistics on the difference using unpooled variance (not allowed if `parmtype!="difference"`)
- method=’score’ - Tstat is a Z statistic formed using score statistics, where the parameter is defined by parmtype, and the constrained maximum likelihood estimates of the parameter are calculated by `constrMLE.difference`, `constrMLE.ratio`, or `constrMLE.oddsratio`.
- method=’FisherAdj’ - Tstat is a one-sided Fisher’s ’exact’ mid p-value. The mid p-value is an adjustment for ties that technically removes the ’exactness’ of the Fisher’s p-value...BUT, here we are only using it to order the sample space, so the results of the resulting unconditional test will still be exact.

- method='user' - Tstat is a user supplied statistic given by `Tfunc`, it must be a function with the first 5 elements of its call being (X1, N1, X2, N2, delta0). The function must returns a vector of length the same as X1 and X2, where higher values suggest larger theta2 compared to theta1 (when `tsmethod!="square"`) or higher values suggest more extreme (when `tsmethod=="square"` and `alternative=="two.sided"`). A slower algorithm that does not require monotonicity of one-sided p-values with respect to delta0 is used.
- method='user-fixed' - For advanced users. Tstat is a user supplied statistic given by `Tfunc`. It should have first 5 elements as described above but its result should not change with delta0 and it must meet Barnard's convexity conditions. If these conditions are met (the conditions are not checked, since checking them will slow the algorithm), then the p-values will be monotonic in delta0 (the null parameter for a two-sided test) and we can use a faster algorithm.

In the basic case, if `alternative="two.sided"`, the argument `tsmethod="central"` gives the two-sided central method. The p-value is just twice the minimum of the one-sided p-values (or 1 if the doubling is greater than 1).

Now consider cases other than the basic case. The `tsmethod="square"` option gives the square of the test statistic (when method="simple", "score", "wald-pooled", or "wald-unpooled") and larger values suggest rejection in either direction (unless method='user', then the user supplies any test statistic for which larger values suggest rejection).

The `tiebreak=TRUE` option breaks ties in a reasonable way when `method="simple"` (see 'details' section of `calcTall`). The `EplusM=TRUE` option performs Lloyd's (2008) E+M ordering on Tstat (see 'details' section of `calcTall`).

If `tiebreak=TRUE` and `EplusM=TRUE`, the tiebreak calculations are always done first.

Berger and Boos (1994) developed a very general method for calculating p-values when a nuisance parameter is present. First, calculate a (1-gamma) confidence interval for the nuisance parameter, check for the supremum over the union of the null hypothesis parameter space and that confidence interval, then add back gamma to the p-value. This adjustment is valid (in other words, applied to exact tests it still gives an adjustment that is exact). The Berger-Boos adjustment is applied when `gamma>0`.

When method='simple' or method='user-fixed' does a simple grid search algorithm using `unirootGrid`. No checks are done on the Tstat function when method='user-fixed' to make sure the simple grid search will converge to the proper answer. So method='user-fixed' should be used by advanced users only.

When `midp=TRUE` the mid p-value is calculated (and the associated confidence interval if `conf.int=TRUE`) instead of the standard p-value. Loosely speaking, the standard p-value calculates the probability of observing equal or more extreme responses, while the mid p-value calculates the probability of more extreme responses plus 1/2 the probability of equally extreme responses. The tests and confidence intervals when `midp=TRUE` are not exact, but give type I error rates and coverage of confidence intervals closer to the nominal values. The mid p-value was studied by Lancaster (1961), see vignette on mid p-values for details.

See Fay and Hunsberger (2021) for a review paper giving the details for these kinds of unconditional exact tests.

Value

The function `uncondExact2x2Pvals` returns a (n1+1) by (n2+1) matrix of p-values for all possible x1 and x2 values, while `uncondExact2x2` returns a list of class 'htest' with elements:

`statistic`	proportion in sample 1
`parameter`	proportion in sample 2
`p.value`	p-value from test
`conf.int`	confidence interval on parameter given by parmtype
`estimate`	MLE estimate of parameter given by parmtype
`null.value`	null hypothesis value of parameter given by parmtype
`alternative`	alternative hypothesis
`method`	description of test
`data.name`	description of data

Warning

The algorithm for calculating the p-values and confidence intervals is based on a series of grid searches. Because the grid searches are often trying to optimize non-monotonic functions, the algorithm is not guaranteed to give the correct answer. At the cost of increasing computation time, better accuracy can be obtained by increasing control$nPgrid, and less often by increasing control$nCIgrid.

Author(s)

Michael P. Fay, Sally A. Hunsberger

References

Berger, R. L. and Boos, D. D. (1994). P values maximized over a confidence set for the nuisance parameter. Journal of the American Statistical Association 89 1012-1016.

Fay, M.P. and Hunsberger, S.A. (2021). Practical valid inferences for the two-sample binomial problem. Statistics Surveys 15:72-110.

Lancaster, H.O. (1961). Significance tests in discrete distributions. JASA 56: 223-234.

Lloyd, C. J. (2008). Exact p-values for discrete models obtained by estimation and maximization. Australian & New Zealand Journal of Statistics 50 329-345.

See Also

See `boschloo` for unconditional exact tests with ordering function based on Fisher's exact p-values.

Examples

```
# default uses method="FisherAdj"
uncondExact2x2(1,10,9,10,
               parmtype="ratio")
uncondExact2x2(1,10,9,10,
               method="score",parmtype="ratio")
```

`uncondPower2x2` *Calculate power or sample size for any 2x2 test.*

Description

The function `Power2x2` and `SS2x2` calculates the power or sample size for any 2x2 test, while the function `uncondPower2x2` calculates power for only tests supported by `uncondExact2x2Pvals`.

Usage

```
Power2x2(n1, n2, theta1, theta2, alpha, pvalFunc, ...)

uncondPower2x2(n1,n2, theta1, theta2, alpha, ...)

SS2x2(theta1, theta2, alpha, pvalFunc, power=0.90,
  n1start=10, increaseby=1, n2.over.n1=1,
  maxiter=50, printSteps=TRUE, ...)
```

Arguments

`n1`	sample size in group 1
`n2`	sample size in group 2
`theta1`	probability of success in group 1
`theta2`	probability of success in group 2
`alpha`	significance level
`pvalFunc`	function that inputs x1,n1,x2,n2 and outputs a p-value.
`power`	target power
`n1start`	value of n1 for first iteration
`increaseby`	positive integer, how much to increase n1 by for each iteration
`n2.over.n1`	ratio of n2/n1
`maxiter`	maximum number of iterations
`printSteps`	logical, should the power and sample size be printed after each iteration?
`...`	arguments passed to `uncondExact2x2Pvals` (for `uncondPower2x2`), or to `Power2x2` (for `SS2x2`). Not used and saved for future use for `Power2x2`.

Details

The function `Power2x2` is a very simple function to calculate power. It calculates power where rejection is when the p-value from `pvalFunc` is less than or equal to `alpha`. The function `SS2x2` repeatedly calls `Power2x2` as it increases the sample size, stopping when the power is greater than `'power'`.

The function `uncondPower2x2` is similar except the p-values are calculated by `uncondExact2x2Pvals`.

Value

the power functions return only the power. The sample size function returns a list of class 'htest.power'.

See Also

For power and sample size for conditional exact tests (e.g., Fisher's exact tests) see `power2x2` and `ss2x2`. For power for the boundary-optimized rejection region (BORR) test see `powerBorr`.

Examples

```
library(exact2x2)
Power2x2(3,4,.1,.9,0.025, pvalFunc=
  function(x1,n1,x2,n2){
      boschloo(x1,n1,x2,n2, alternative="greater",
        or=1,tsmethod="central", midp=TRUE)$p.value
  }
)
##
## Not run:
SS2x2(.1,.9,0.025, n1start=5, pvalFunc=
           function(x1,n1,x2,n2){
             boschloo(x1,n1,x2,n2, alternative="greater",
                      or=1,tsmethod="central", midp=TRUE)$p.value
           }
)

## End(Not run)
```

`unirootGrid` *Function to find a root by grid search.*

Description

Find the root (value where the function equals 0) of a monotonic function, `func`, using a halving algorithm grid search.

Usage

```
unirootGrid(func, power2 = 12, step.up = TRUE, pos.side = FALSE,
    print.steps = FALSE, power2grid = power2gridRatio, ...)
```

Arguments

`func`	monotonic function
`power2`	positive integer, number of grid points is 1+2^power2
`step.up`	logical, start the search at the lower end of the grid and step up?
`pos.side`	logical, should the root be on the positive side? In other words, should func(root)>=0

`print.steps`	logical, should each step that is evaluated be printed?
`power2grid`	function that returns the grid. Take one argument, `power2`
`...`	arguments passed to `func`

Details

The grid is defined with the `power2grid` argument that defines a function with an argument `power2`, and returns a grid with `1+2^power2` elements. The root is found by a halving algorithm on the grid, so `func` is calculated only `power2+1` times. The 'root' is the element that is closest to the root, either on the positive side (`pos.side=TRUE`) or not.

The `unirootGrid` function calls `uniroot.integer` and finds roots based on grid search. The functions `power2gridRatio` and `power2gridDifference` create grids for searching (0,Inf) and (-1,1) respectively. The `power2gridRatio` grid is equally spaced on the log scale with about half of the grid between 0.5 and 2. The function `power2grid` allows more flexibility in defining grids.

Value

A list with elements:

`iter`	number of iterations
`f.root`	value of func at root
`root`	root, element on the grid that is closest to the root on the negative side (if pos.side=FALSE)
`bound`	interval for the accuracy

Author(s)

Michael P. Fay

See Also

`uniroot` and `uniroot.integer`

Examples

```
# print.steps prints all iterations,
# with x=rank of grid value (e.g., x=1 is lowest value in grid)
# f(x) really is f(grid[x]) where grid is from the power2grid function
unirootGrid(function(x){ x - .37}, power2=10, power2grid=power2gridRatio,
  print.steps=TRUE, pos.side=TRUE)
```

Index