Package 'binomSamSize'

December 5, 2016

Type Package

Title Confidence Intervals and Sample Size Determination for a Binomial Proportion under Simple Random Sampling and Pooled Sampling

Version 0.1-4

Date 2016-12-04

Author Michael Höhle [aut, cre], Wei Liu [ctb]

Maintainer Michael Höhle <hoehle@math.su.se>

Depends binom

Suggests testthat

Description A suite of functions to compute confidence intervals and necessary sample sizes for the parameter p of the Bernoulli B(p) distribution under simple random sampling or under pooled sampling. Such computations are e.g. of interest when investigating the incidence or prevalence in populations.
The package contains functions to compute coverage probabilities and coverage coefficients of the provided confidence intervals procedures. Sample size calculations are based on expected length.

License GPL-3

LazyLoad yes

Encoding latin1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-12-05 08:27:24

R topics documented:

`binomSamSize-package` *Confidence intervals and sample size determination for a binomial proportion under simple random sampling and pooled sampling*

Description

A suite of functions to compute confidence intervals and necessary sample sizes for the parameter p of the Bernoulli $B(p)$ distribution under simple random sampling or under pooled sampling. Such computations are e.g. of interest when investigating the incidence or prevalence in populations.

The package contains functions to compute coverage probabilities and coverage coefficients of the provided confidence intervals procedures. Sample size calculations are based on expected length and coverage probabilities of the resulting confidence intervals.

Details

Package:	binomSamSize
Type:	Package
Version:	0.1-3
Date:	2013-12-10
License:	GPL-3
LazyLoad:	yes

The packages relies heavily on the functions in the package `binom` by Sundar Dorai-Raj.

Other R packages containing similar functionality for calculating sample sizes for pooled sampling is the `binGroup` package. Also the `epiR` has further functionality for computing sample sizes for tests with sensitivity and specificity different from 1.

Functionality for pooled prevalence calculations with a nice web interface can also be found at http://www.ausvet.com.au/pprev/content.php?page=home

Note that the package is still in a development stage. As a consequence, use the results of the package with care because bugs, inconsistencies and errors might exist. In case you find bugs please do not hesitate to report them to the package maintainer.

Author(s)

Michael Höhle with contributions by Wei Liu

Maintainer: Michael Höhle <hoehle@math.su.se>

References

Piegorsch, W. W. (2004), Sample sizes for improved binomial confidence intervals, Computational Statistics and Data Analysis, 46:309–316.

See Also

Package `binom.confint{binom::binom.confint}`

`binom.liubailey` *Calculate fixed width confidence interval for binomial proportion*

Description

Calculate a fixed width confidence interval for the the binomial proportion based on one observation from the binomial distribution

Usage

```
binom.liubailey(x, n, d, lambda=0, conf.level=0.95)
```

Arguments

`x`	Vector of number of successes in the binomial experiment.
`n`	Vector of number of independent trials in the binomial experiment.
`conf.level`	The level of confidence to be used in the confidence interval
`d`	half width of the confidence interval
`lambda`	Shrinkage factor. `lambda=0` corresponds to simple $\hat{p} \pm d$ interval.

Details

The confidence interval is as suggested in equation (3.1) of Liu & Bailey (2002).

$$(\hat{p}_l, \hat{p}_u) = (C_n(\hat{p}_n) - d, C_n(\hat{p}_n) + d)$$

The exact form is as follows: Let z be the appropriate $(1 - \texttt{conf.level})/2$ quantile of the standard normal distribution the interval with shrinkage towards 0.5 is given by:

$$(\hat{p}_l, \hat{p}_u) = \hat{p}_n + \frac{\lambda z^2 (0.5 - \hat{p}_n)}{n + z^2} \pm d$$

The interval is then expanded to a full length of $2d$ using the following transformation:

$$\hat{p}_l^* = \max(0, \min(1 - 2d, \hat{p}_l))$$

$$\hat{p}_u^* = \min(1, \max(2d, \hat{p}_u))$$

As a consequence, the computed interval will always have length $2d$.

If fixed length is a desired property of your CI then this is a way to go. However, the Liu and Bailey (2002) confidence intervals can have a low coverage coefficients when n is very small compared to d. When using the sample size computation procedure in `ciss.liubailey` one however ensures that n is large enough for the selected d to guarantee the required coverage coefficient. Thus, one should use `binom.liubailey` in connection with `ciss.liubailey`.

Value

A data.frame containing the observed proportions and the lower and upper bounds of the confidence interval. The style is similar to the `binom.confint` function of the `binom` package

Author(s)

M. Höhle

References

Liu, W. and Bailey, B.J.R. (2002), Sample size determination for constructing a constant width confidence interval for a binomial success probability. Statistics and Probability Letters, 56(1):1-5.

See Also

`ciss.liubailey`

Examples

```
binom.liubailey(x=0:20,n=20, d=0.1, lambda=0)

#Compute coverage of this interval
cov <- coverage( binom.liubailey, n=20, alpha=0.05, d=0.1, lambda=0,
                 p.grid=seq(0,1,length=1000))

plot(cov,type="l")
```

---

binom.midp *Calculate mid-p confidence interval for binomial proportion*

Description

Calculate mid-p confidence interval for the the binomial proportion based on one observation from the binomial distribution

Usage

```
binom.midp(x, n, conf.level=0.95)
```

Arguments

x	Vector of number of successes in the binomial experiment.
n	Vector of number of independent trials in the binomial experiment.
conf.level	The level of confidence to be used in the confidence interval

Details

The function uses uniroot to determine the upper and lower bounds of the mid-p confidence interval.

The lower bound p_l is found as the solution to the equation

$$\frac{1}{2}f(x; n, p_l) + (1 - F(x; m, p_l)) = \frac{\alpha}{2}$$

where $f(x; n, p)$ denotes the probability mass function (pmf) and $F(x; n, p)$ the (cumulative) distribution function of the binomial distribution with size n and proportion p evaluated at x. In case x=0 then the lower bound is zero.

The upper bound p_u is found as the solution to the equation

$$\frac{1}{2}f(x; n, p_u) + F(x - 1; m, p_u) = \frac{\alpha}{2}$$

In case x=n then the upper bound is 1.

Value

A data.frame containing the observed proportions and the lower and upper bounds of the confidence interval. The style is similar to the binom.confint function of the binom package

Author(s)

M. Höhle

References

S. E. Vollset (1993), Confidence intervals for a binomial proportion, Statistics in Medicine, 12, 809–824

Fosage, G.T. (2005) Modified exact sample size for a binomial proportion with special emphasis on diagnostic test parameter estimation, Statistics in Medicine 24(18):2857-66.

A. Agresti and A. Gottard (2005), Comment: Randomized Confidence Intervals and the Mid-P Approach, Statistical Science, 20(4):367–371

Examples

```
binom.midp(x=0:10,n=10)
binom.midp(x=0:5,n=5,conf.level=0.9)
```

`ciss.binom` *General purpose sample size calculation based on confidence interval widths*

Description

Calculate necessary sample size for estimating a binomial proportion with the confidence interval computed by an arbitrary `binom.confint` function

Usage

```
ciss.binom(p0, d, alpha=0.05, ci.fun=binom.confint,
           np02x = function(n, p0) round(n*p0), verbose=FALSE,
           nStart=1,nMax=1e6,...)
```

Arguments

`p0`	hypothesized value of the parameter p in the binomial distributionproportion. This is an upper bound if `p0` is below 1/2, and a lower bound if `p0` is above 1/2.
`d`	half width of the confidence interval. Note: The CI is not necessarily symmetric about the estimate so we just look at its width as determine by $d = 1/2 * (CI_upper - CI_lower)$.
`alpha`	a two-sided $(1 - \alpha) \cdot 100\%$ confidence interval is computed
`ci.fun`	Any `binom.confint` like confidence interval computing function. The default is the `binom.confint` function itself. In this case one would have to specify the appropriate method to use using the `method` argument of the `binom.confint` function.
`np02x`	A function specifying how to calculate the value of x which results in an estimator of the proportion being as close as possible to the anticipated value p_0. Typically the value is obtained by rounding the result of $x \cdot p_0$.
`verbose`	If `TRUE`, additional output of the computations are shown. The default is `FALSE`.
`nStart`	Value where to start the search. The default `n=1` can sometimes lead to wrong answers, e.g. for the Wald-type interval
`nMax`	Max value of the sample size n to try in the iterative search. See details
`...`	Additional arguments sent to `ci.fun` function

Details

Given a pre set α-level and an anticipated value of p, say p_0, the objective is to find the minimum sample size n such that the confidence interval will lead to an interval of length $2 \cdot d$.

Using `ciss.binom` this is done in a general purpose way by performing an iterative search for the sample size. Starting from $n = nStart$ the appropriate x value, computed as `round(x*p0)`, is found. For this integer x and the current n the corresponding confidence interval is computed using the function `ci.fun`. This function has to deliver the same type of result as the `binom.confint`

function, i.e. a data frame containing the arguments `lower` and `upper` containing the borders of the confidence interval.

The sample size is iteratively increased until the obtained confidence interval has a length smaller than $2 \cdot d$. This might take a while if n is large. It is possible to speed up the search if an appropriate `nStart` is provided.

A brute force search is used within the function. Note that for many of the confidence intervals explicit expressions exists to calculate the necessary sample size.

Value

the necessary sample size n

Author(s)

M. Höhle

See Also

`binom.confint` and its related functions

Examples

```
#Compute the classical Wald-type interval using brute force search
#Note that nStart=2 needs to be called, because the Wald-intervals
#for x=round(1*0.5)=0 is too short.
ciss.binom(p0=1/2, d=0.1, alpha=0.05, method="asymptotic",nStart=2)
#This could of course be done easier
ciss.wald(p0=1/2, d=0.1, alpha=0.05)

#Same for the Wilson intervals
ciss.binom(p0=1/2, d=0.1, alpha=0.05, method="wilson")
ciss.wilson(p0=1/2, d=0.1, alpha=0.05)

#Now the mid-p intervals
ciss.binom(p0=1/2, d=0.1, alpha=0.05, ci.fun=binom.midp)
#This search in Fosgate (2005) is a bit different, because interest
#is not directly in the length, but the length is used to derive
#the upper and lower limits and then a search is performed until
#the required alpha level is done. The difference is negliable
ciss.midp(p0=1/2, d=0.1, alpha=0.05)

#Another situation where no closed formula exists
ciss.binom(p0=1/2, d=0.1, alpha=0.05, method="lrt")

#Pooled samples. Now np02x is a func taking three arguments
#The k argument is provided as additional argument
np02x <- function(n,p0,k) round( (1-(1-p0)^k)*n )
ciss.binom( p0=0.1, d=0.05, alpha=0.05, ci.fun=poolbinom.lrt,
            np02x=np02x, k=10,verbose=TRUE)
```

`ciss.liubailey` *Sample size calculations using the Liu and Bailey (2002) approach*

Description

Calculate sample size for a binomial proportion based on the confidence interval width specification in Liu and Bailey (2002).

Usage

```
ciss.liubailey(alpha, d, lambda.grid = 0:30)
```

Arguments

`alpha`	a $(1-\alpha/2)\cdot 100\%$ confidence interval is computed
`d`	half width of the confidence interval
`lambda.grid`	range of lambda values to try

Details

The objective is to find the minimum sample size n so that the minimum coverage probability (aka. as the coverage coefficient) of the confidence interval for the binomial parameter is larger than $1-\alpha$. In the present approach the confidence interval is of form

$$(C_n(\hat{p}_n) - d, C_n(\hat{p}_n) + d)$$

as suggested in equation (3.1) of Liu & Bailey (2002):

$$(\hat{p}_l, \hat{p}_u) = \hat{p}_n + \frac{\lambda z^2 (0.5 - \hat{p}_n)}{n + z^2} \pm d$$

where $\hat{p}_n = x/n$. The interval is then expanded to a full length of $2d$ using the following transformation:

$$\hat{p}_l^* = \max(0, \min(1 - 2d, \hat{p}_l))$$
$$\hat{p}_u^* = \min(1, \max(2d, \hat{p}_u))$$

As a consequence, the computed interval will always have length $2d$.

Given `d`, fixed `lambda` and a sample size `n`, the proportion p in [0,1] where the coverage probability is minimum is computed. The sample size is then gradually increased until this minimum coverage probability becomes larger than $1-\alpha$. We then change the value of λ, and search the minimum sample size that guarantee the $1-\alpha$ confidence level for this lambda value. The smallest minimum sample size over a set of lambda values in `lambda.grid` is then used as the required sample size; this sample size and the corresponding lambda value are used to calculate the confidence interval given above.

For a general overview of coverage probabilities of confidence intervals for a binomial proportion see Agresti and Coull (1998). Once actual binomial data are obtained the function `binom.liubailey` can be used to compute the actual confidence interval.

The R function code calls the original Fortran code developed for the Liu and Bailey (2002) article. NAG calls were replaced by R API calls and an R wrapper calling the code as a subroutine was created.

Value

a vector containing the following three elements

nstar	sample size at most favorable lambda value in lambda.grid
cp	coverage probability
lambda	value in lambda.grid giving the lowest nstar value

Author(s)

M. Höhle and W. Liu

References

Agresti, A. and Coull, B.A. (1998), Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions, The American Statistician, 52(2):119-126.

Liu, W. and Bailey, B.J.R. (2002), Sample size determination for constructing a constant width confidence interval for a binomial success probability. Statistics and Probability Letters, 56(1):1-5.

See Also

binom.liubailey

Examples

```
ciss.liubailey(alpha=0.1,d=0.05)
ciss.liubailey(alpha=0.1,d=0.05,lambda.grid=5)
```

---

ciss.midp *Sample size calculations using the Fosgate (2005) approach*

Description

Calculate sample size for a binomial proportion based on a mid-p confidence interval width specification.

Usage

```
ciss.midp(p0, d, alpha, nMax=1e6)
```

Arguments

p0	hypothesized upper bound (if below 0.5, if above 0.5 then lower bound) on the parameter p in the binomial distribution
alpha	an $(1-\alpha/2)\cdot 100\%$ confidence interval is computed
d	half width of the confidence interval
nMax	Largest n to check. Interrupt iterations when this value is reached

Details

Fosgate (2005) discusses the need for improved sample size calculations in cases where the binomial proportion is close to 0 and 1. To improve on this, calculation on confidence intervals based on the mid-p method are suggested where computation of the upper and lower limit are combined into one formula. Given lower and upper bounds p_l and p_u of the (1-`alpha`)*100% confidence interval, one finds the sample size n as the solution to

$$\frac{1}{2}f(x;n,p_l) + \frac{1}{2}f(x;n,p_u) + (1 - F(x;n,p_l)) + F(x-1;m,p_u) = \alpha$$

where $f(x;n,p)$ denotes the probability mass function (pmf) and $F(x;n,p)$ the (cumulative) distribution function of the binomial distribution with size n and proportion p evaluated at x. The function then returns $\lceil n \rceil$. Note that in this approach $(p_l, p_u) = p_0 \pm d$, which has to be a subset of $(0,1)$. Another option would be to choose the lower and upper independent specifically.

In the above, x is found as the integer value, such that x/n is as close as possible to the hypothesized value `p0` as possible.

An alternative approach to determine sample sizes based on the mid-p approach is to manually find the sample size n such that the interval obtained by `binom.midp` has a length less than $2 \cdot d$.

Value

the necessary sample size n

Author(s)

M. Höhle

References

Fosage, G.T. (2005) Modified exact sample size for a binomial proportion with special emphasis on diagnostic test parameter estimation, Statistics in Medicine 24(18):2857-66.

See Also

`binom.midp`, `ciss.binom`

Examples

```
#Fosgate approach
ciss.midp(p0=0.2,alpha=0.05,d=0.05)
#Iterative increase of n using the general purpose function
ciss.binom( p0=0.2, alpha=0.05, ci.fun=binom.midp, d=0.05)
```

`ciss.pool.wald` *Sample size calculations for fixed pool size and perfect test*

Description

Calculate sample size for a binomial proportion based on Wald type confidence interval for pooled samples using a perfect test

Usage

```
ciss.pool.wald(pi0, alpha, d, k)
```

Arguments

`pi0`	hypothesized upper bound (if below 0.5, if above 0.5 then lower bound) on the parameter π_0
`alpha`	an $(1-\alpha/2)\cdot 100\%$ confidence interval is computed
`d`	half width of the confidence interval
`k`	The pool size

Details

Sample size calculation based on width of Wald confidence intervals for pooled sample. The equation is

$$n = \left\lceil \left(\frac{z_{1-\alpha/2}(1-\pi_0)}{dk} \right)^2 \cdot \left((1-\pi_0)^{-k} - 1 \right) \right\rceil$$

Value

the necessary sample size n

Author(s)

M. Höhle

References

D. D. Worlund and G. Taylor (1983), Estimation of Disease Incidence in Fish Populations, Can. J. Fish. Aquat. Sci., 40:2194-2197.

See Also

`poolbinom.wald`

Examples

```
k <- 1:50
n <- sapply(k, function(k) ciss.pool.wald(pi0=0.1, alpha=0.05,k=k, d=0.05))
#sizes <- cbind(k=k, n=n, N=n*k)
plot(k, n, type="s",xlab="Pool size",ylab="Number of pools")
plot(k*n,n,type="s",xlab="Total size",ylab="Number of pools")

ciss.pool.wald(pi0=0.1, d=0.01, alpha=0.05, k=10)
#Compare with ciss.binom function
np02x <- function(n,p0,k) round( (1-(1-p0)^k)*n )
(n <- ciss.binom( p0=0.1, d=0.01, alpha=0.05, ci.fun=poolbinom.wald,
            np02x=np02x, k=10))
```

`ciss.wilson` *Sample sizes for improved binomial confidence intervals*

Description

Calculate sample size for a binomial parameter enhancing the traditional Wald-type interval

Usage

```
ciss.wald(p0, d, alpha)
ciss.wilson(p0, d, alpha)
ciss.agresticoull(p0, d, alpha)
```

Arguments

`p0`	hypothesized upper bound (if below 0.5, if above 0.5 then lower bound) on the parameter p in the binomial distribution
`alpha`	an $(1-\alpha/2)\cdot 100\%$ confidence interval is computed
`d`	half width of the confidence interval

Details

Given a pre set α-level and an anticipated value of p, say p_0, the objective is to find the minimum sample size n such that the confidence interval will lead to an interval of length $2 \cdot d$.

The work in Piegorsch (2004) gives a number of formulas enhancing the traditional Wald-type interval.

Value

the necessary sample size n

Author(s)

M. Höhle

References

Piegorsch, W. W. (2004), Sample sizes for improved binomial confidence intervals, Computational Statistics and Data Analysis, 46:309–316.

See Also

ciss.midp

Examples

```
#Simple calculation at one proportion (worst case)
ciss.wald(p0=0.5,alpha=0.1,d=0.05)

#Evaluate for a grid of hypothesized proportion
p.grid <- seq(0,0.5,length=100)
cissfuns <- list(ciss.wald, ciss.wilson, ciss.agresticoull)
ns <- sapply(p.grid, function(p) {
  unlist(lapply(cissfuns, function(f) f(p, d=0.1, alpha=0.05)))
})

matplot(p.grid, t(ns),type="l",xlab=expression(p[0]),ylab="n",lwd=2)
legend(x="topleft", c("Wald", "Wilson","Agresti-Coull"), col=1:3, lty=1:3,lwd=2)
```

---

coverage *Calculate coverage probability for a binomial proportion confidence interval scheme*

Description

For a given true value of the proportion compute the coverage probability of the confidence interval

Usage

```
coverage(ci.fun, n, alpha=0.05, p.grid=NULL,interval=c(0,1),
         pmfX=function(k,n,p) dbinom(k,size=n,prob=p), ...)
## S3 method for class 'coverage'
plot(x, y=NULL, ...)
```

Arguments

ci.fun	binom.confint like function which computes confidence intervals for a binomial proportion.
n	Sample size of the binomial distribution.
alpha	Level of significance, $1-\alpha$ is the confidence level.

p.grid	Vector of proportions where to evaluate the confidence interval function. If NULL all those values where the minimum coverage probability can occur is taken. If not NULL then the union between p.grid and these values is taken.
interval	Vector of length two specifying lower and upper border of an interval of interest for the proportion. The intersection of the above grid and this interval is taken.
pmfX	A function based on the arguments k, n and p, giving the probability mass function (pmf) $f(x; n, p) = P(X = k; n, p)$ of X. Typically, this will be the pmf of the binomial distribution.
x	An object of class coverage
y	Not used
...	Further arguments to be sent to ci.fun or the plot function

Details

Compute coverage probabilities for each proportion in p.grid. See actual function code for the exact details, which p.grid is actually chosen.

Value

An object of class coverage containing coverage probabilities, coverage coefficient and more.

Author(s)

M. Höhle

References

Agresti, A. and Coull, B.A. (1998), Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions, The American Statistician, 52(2):119-126.

Examples

```
#Show coverage of Liu & Bailey interval
cov <- coverage( binom.liubailey, n=100, alpha=0.05,
           p.grid=seq(0,1,length=1000), interval=c(0,1), lambda=0, d=0.1)
plot(cov, type="l")

#Now for some more advanced stuff. Investigate coverage of pooled
#sample size estimators
kk <- 10
nn <- 20
ci.funs <- list(poolbinom.wald, poolbinom.logit, poolbinom.lrt)
covs <- lapply( ci.funs, function(f) {
 coverage( f, n=nn, k=kk, alpha=0.05, p.grid=seq(0,1,length=100),
pmfX=function(k,n,p) dbinom(k,size=n, p=1-(1-p)^kk))
})

par(mfrow=c(3,1))
plot(covs[[1]],type="l",main="Wald",ylim=c(0.8,1))
lines(c(0,1),rep(0.95,2),lty=2,col=2)
```

```
plot(covs[[2]],type="l",main="Logit")#,ylim=c(0.8,1))
lines(c(0,1),rep(0.95,2),lty=2,col=2)
plot(covs[[3]],type="l",main="LRT",ylim=c(0.8,1))
lines(c(0,1),rep(0.95,2),lty=2,col=2)

poolbinom.wald(x=1,n=nn,k=kk)
poolbinom.logit(x=1,n=nn,k=kk)
poolbinom.lrt(x=1,n=nn,k=kk)
```

`poolbinom.logit` *Calculate logit based confidence interval for binomial proportion for pooled samples*

Description

Calculate logit based confidence interval for the the Bernoulli proportion of $k \cdot n$ individuals, which are pooled into n pools each of size k. Observed is the number of positive pools x.

Usage

```
poolbinom.wald(x, k, n, conf.level=0.95)
poolbinom.logit(x, k, n, conf.level=0.95)
```

Arguments

`x`	Number of positive pools (can be a vector).
`k`	Pool size (can be a vector).
`n`	Number of pools (can be a vector).
`conf.level`	The level of confidence to be used in the confidence interval

Details

Assume the individual probability of experiencing the event for each of $k \cdot n$ individuals is π, i.e. the response is Bernoulli distributed $X_i \sim B(\pi)$. For example π could be the prevalence of a disease in veterinary epidemiology.

Now, instead of considering each individual the $k \cdot n$ samples are pooled into n pools each of size k. A pool is positive if there is at least one positive in the pool. Let X be the number of positive pools. Then

$$X \sim Bin(n, 1 - (1 - \pi)^k)$$

.

The present function computes an estimator and confidence interval for π by computing the MLE and standard error for $\hat{\pi}$. A Wald confidence interval is formed using $\hat{\pi} \pm z_{1-\alpha/2} \cdot se(\hat{\pi})$. In case of `poolbinom.logit` a logit transformation is used, i.e. the standard error for $logit(\hat{\pi})$ is computed and the Wald-CI is derived on the logit-scale which is then backtransformed using the inverse logit function. In case $x = 0$ or $x = n$ the logit of $\hat{\pi}$ is not defined and hence the confidence interval

is not defined in these two situation. To fix the problem we use the intervals $(0, \hat{\pi}_u(x = 0))$ and $(\hat{\pi}_l(x = n), 1)$, respectively, where π_u and π_o are the respective borders of a corresponding LRT interval.

The `poolbinom.wald` approach corresponds to method 2 in the Cowling et al. (1999). The logit transformation improves on this procedure, because the method ensures that the interval is in the range (0,1).

Value

A data.frame containing the observed proportions and the lower and upper bounds of the confidence interval. The style is similar to the `binom.confint` function of the `binom` package

Author(s)

M. Höhle

References

D. W. Cowling, I. A. Gardner, W. O. Johnson (1999), Comparison of methods for estimation of individual level prevalence based on pooled samples, Preventive Veterinary Medicine, 39:211–225

See Also

`poolbinom.lrt`

Examples

```
poolbinom.wald(x=0, k=10, n=34, conf.level=0.95)
poolbinom.logit(x=0:1, k=10, n=34, conf.level=0.95)
poolbinom.logit(x=1, k=seq(10,100,by=10), n=34, conf.level=0.95)
poolbinom.logit(x=0:34,k=1,n=34)
```

`poolbinom.lrt` *Calculate LRT based confidence interval for binomial proportion for pooled samples*

Description

Calculate LRT based confidence interval for the Bernoulli proportion of $k \cdot n$ individuals, which are pooled into n pools each of size k. Observed is the number of positive pools x.

Usage

```
poolbinom.lrt(x, k, n, conf.level=0.95, bayes=FALSE, conf.adj=FALSE)
```

Arguments

x	Number of positive pools (can be a vector).
k	Pool size (can be a vector).
n	Number of pools (can be a vector).
conf.level	The level of confidence to be used in the confidence interval
bayes	See binom.bayes
conf.adj	See binom.bayes

Details

Compute LRT based intervals for the binomial response $X \sim Bin(n, \theta)$, where $\theta = 1 - (1 - \pi)^k$. As a consequence,

$$\pi = g(\theta) = 1 - (1 - \pi)^{1/k}$$

.

One then knows that the borders for π are just transformations of the borders of theta using the above $g(\theta)$ function.

For further details about the pooling setup see poolbinom.logit.

Value

A data.frame containing the observed proportions and the lower and upper bounds of the confidence interval. The output is similar to the binom.confint function of the binom package

Author(s)

M. Höhle

Examples

```
binom.lrt(x=0:34,n=34)
poolbinom.lrt(x=0:34,k=1,n=34)
poolbinom.lrt(x=0:34,k=10,n=34)
```

Index