Package 'binGroup'

July 22, 2025

Title Evaluation and Experimental Design for Binomial Group Testing

Version 2.2-3

Date 2025-05-07

Maintainer Frank Schaarschmidt <schaarschmidt@biostat.uni-hannover.de>

Imports graphics, partitions, Rdpack, stats, utils

RdMacros Rdpack

Description Methods for estimation and hypothesis testing of proportions in group testing designs: methods for estimating a proportion in a single population (assuming sensitivity and specificity equal to 1 in designs with equal group sizes), as well as hypothesis tests and functions for experimental design for this situation. For estimating one proportion or the difference of proportions, a number of confidence interval methods are included, which can deal with various different pool sizes. Further, regression methods are implemented for simple pooling and matrix pooling designs.
Methods for identification of positive items in group testing designs: Optimal testing configurations can be found for hierarchical and array-based algorithms. Operating characteristics can be calculated for testing configurations across a wide variety of situations.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Boan Zhang [aut],
Christopher Bilder [aut],
Brad Biggerstaff [aut],
Frank Schaarschmidt [aut, cre],
Brianna Hitt [aut]

Repository CRAN

Date/Publication 2025-05-14 12:20:02 UTC

Contents

`binGroup-package` *Statistical Methods for Group Testing.*

Description

This package provides methods for estimation and hypothesis testing of proportions in group testing designs. It involves methods for estimating a proportion in a single population (assuming sensitivity and specificity 1 in designs with equal group sizes), as well as hypothesis tests and functions for experimental design for this situation. For estimating one proportion or the difference of proportions, a number of confidence interval methods are included, which can deal with various different pool sizes. Further, regression methods are implemented for simple pooling and matrix pooling designs.

This package also provides methods for identification of positive items in a number of group testing algorithms. Optimal testing configurations can be found for hierarchical and array-based algorithms. Operating characteristics can be calculated for testing configurations across a wide variety of situations.

Details

Package:	binGroup
Type:	Package
Version:	2.2-1
Date:	2018-07-07
License:	GPL (>=3)
LazyLoad:	no

1) One-sample case

Methods for calculating confidence intervals for a single population proportion from designs with equal group sizes (as described by Tebbs and Bilder, 2004 and Schaarschmidt, 2007) are implemented in the function `bgtCI`.

For the problem of choosing an adequate experimental design in the one-sample case with only one group size, the functions estDesign, sDesign, nDesign implement different iterative approaches, as examplified by Swallow (1985) and Schaarschmidt (2007).

If a confidence interval for a single proportion shall be calculated based on a design involving groups of different group sizes, a number of methods described by Hepworth (1999) is available in the function pooledBin. The exact method described by Hepworth (1996) is implemented in the function bgtvs.

2) Two-sample case

The function pooledBinDiff provides a number of confidence interval methods for estimating the difference of proportions from two independent samples, allowing for groups of different group size (Biggerstaff, 2008).

3) Regression models

Two approaches (by Vansteelandt et al., 2000 and Xie, 2001) to estimate parameters of group testing regression models can be applied by calling gtreg. Once fitted, corresponding methods to extract residuals, calculate predictions and summarize the parameter estimates (including hypotheses tests) are available in the S3 methods residuals.gt, predict.gt and summary.gt.

Group testing regression models in settings with matrix pooling (Xie, 2001) can be fit using gtreg.mp.

4) Identification using hierarchical and array-based group testing algorithms

The function OTC implements a number of group testing algorithms, described in Hitt et al. (2018), which calculate the operating characteristics and find the optimal testing configuration over a range of possible initial pool sizes and/or testing configurations (sets of subsequent pool sizes).

Author(s)

Boan Zhang, Christopher Bilder, Brad Biggerstaff, Brianna Hitt, Frank Schaarschmidt

Maintainer: Frank Schaarschmidt <schaarschmidt@biostat.uni-hannover.de>

References

Biggerstaff, B.J. (2008): Confidence interval for the difference of proportions estmimated from pooled samples. *Journal of Agricultural Biological and Environmental Statistics, 13(4), 478-496.*

Hepworth, G. (1996) Exact confidence intervals for proportions estimated by group testing. *Biometrics 52, 1134-1146.*

Hepworth, G. (1999): *Estimation of proportions by group testing.* PhD Dissertation. Melbourne, Australia: The University of Melbourne.

Hitt, B., Bilder, C., Tebbs, J. & McMahan, C. (2018) The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!. Manuscript submitted for publication. http://www.chrisbilder.com/grouptesting

Schaarschmidt, F. (2007) Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. *Communications in Biometry and Crop Science 2 (1), 32-40.* http://agrobiol.sggw.waw.pl/cbcs/

Swallow, W.H. (1985) Group testing for estimating infection rates and probabilities of disease transmission. *Phytopathology 75 (8), 882-889.*

Tebbs, J.M. & Bilder, C.R. (2004) Confidence interval procedures for the probability of disease transmission in multiple-vector-transfer designs. *Journal of Agricultural, Biological and Environmental Statistics 9 (1), 75-90.*

Vansteelandt, S., Goetghebeur, E., and Verstraeten, T. (2000) Regression models for disease prevalence with diagnostic tests on pools of serum samples, *Biometrics, 56, 1126-1133.*

Xie, M. (2001) Regression analysis of group testing samples, *Statistics in Medicine, 20, 1957-1969.*

Examples

```
# 1) One-sample problem

# 1.1) Confidence intervals for designs with equal group size (pool size),
# where
#  n denotes the number of groups (pools),
#  s denotes the common group size (number of individuals pooled per group),
#  y denotes the number of groups tested positive.

# The following call reproduces the example given
# by Tebbs and Bilder (2004) for the two-sided 95-percent
# exact (Clopper-Pearson) interval:

bgtCI(n=24, y=3, s=7, conf.level=0.95,
 alternative="two.sided", method="CP")

# 1.2) Confidence intervals for designs with unequal group size (pool size):
# Keeping notation as above but allowing for (a limited number of) different
# group size s, the examples given in Hepworth (1996), Table 5 can be
# reproduced by calling:

 bgtvs(n=c(2,3), s=c(5,2), y=c(0,0))
 bgtvs(n=c(2,3), s=c(5,2), y=c(0,1))

# The function pooledBin provides different methods for the same problem,
# where x is the number of positive groups, m is the size of the groups and
# n is the number of groups with the correesponding sizes:

pooledBin(x=c(0,1), m=c(5,2), n=c(2,3), ci.method="score")
pooledBin(x=c(0,1), m=c(5,2), n=c(2,3), ci.method="lrt")
pooledBin(x=c(0,1), m=c(5,2), n=c(2,3), ci.method="bc-skew-score")

# 1.3) For experimental design based on the bias of the point estimate,
# as proposed by Swallow (1985): The values in Table 1 (Swallow, 1985),
# p.885 can be reproduced by calling:

estDesign(n=10, smax=100, p.tr=0.001)
estDesign(n=10, smax=100, p.tr=0.01)

# 2) Two-sample comparison

# Assume a design, where pools 5, 1, 1, 30, and 20 pools of size 10, 4, 1, 25, 50,
# respectively, are used to estimate the prevalence in two populations.
# In population 1, one out of 5 pools with 10 units is positive,
```

```
# while in population 2, two out of five pools with 10 units is positive as well as
# the one pool with only 1 unit.
# The difference of proportions is to be estimated.

x1 <- c(1,0,0,0,0)
m1 <- c(10,4,1,25,50)
n1 <- c(5,1,1,30,20)

x2 <- c(2,0,1,0,0)
m2 <- c(10,4,1,25,50)
n2 <- c(5,1,1,30,20)

pooledBinDiff(x1=x1, m1=m1,x2=x2, m2=m2, n1=n1, n2=n2, ci.method="lrt")


# 3) Regression models

# 3.1) Fitting a regression model
# A HIV surveillance data (used by Vansteelandt et al. 2000)
# can be analysed for the dependence of HIV prevalence
# on covariates AGE and EDUC., with sensitivity and specificity
# assumed to be 0.9 each.

data(hivsurv)
fit1 <- gtreg(formula = groupres ~ AGE + EDUC., data = hivsurv,
  groupn = gnum, sens = 0.9, spec = 0.9, method = "Xie")
summary(fit1)


# 3.2) Fitting a regression model for matrix pooling data
# The function sim.mp is used to simulate a matrix pooling data set:

set.seed(9128)
sa1a<-sim.mp(par=c(-7,0.1), n.row=c(5,4), n.col=c(6,5),
 sens=0.95, spec=0.95)

str(sa1a)
sa1<-sa1a$dframe


## Not run:
fit2 <- gtreg.mp(formula = cbind(col.resp, row.resp) ~ x, data = sa1,
                 coln = coln, rown = rown, arrayn = arrayn,
                 sens = 0.95, spec = 0.95, n.gibbs = 2000, trace = TRUE)
fit2
summary(fit2)

## End(Not run)

# 4) Identification using hierarchical and array-based group testing algorithms

# 4.1) Finding the optimal testing configuration over a range of initial
# group sizes, using non-informative three-stage hierarchical testing, where
```

```
#   p denotes the overall prevalence of disease,
#   Se denotes the sensitivity of the diagnostic test,
#   Sp denotes the specificity of the diagnostic test,
#   group.sz denotes the range of initial pool sizes for consideration, and
#   obj.fn specifies the objective functions for which to find results.

# The following call reproduces results given by Hitt et al. (2018) for
# informative three-stage hierarchical testing with an overall disease
# prevalence of E(p_i) = 0.01 and sensitivity and specificity equal to 0.95.

# This example takes approximately 2.5 minutes to run.

## Not run:
set.seed(1002)
results1 <- OTC(algorithm="ID3", p=0.01, Se=0.95, Sp=0.95, group.sz=3:40,
obj.fn=c("ET", "MAR"), alpha=2)
results1$opt.ET$OTC
results1$opt.ET$ET/results1$opt.ET$OTC$Stage1
results1$opt.MAR$OTC
results1$opt.MAR$ET/results1$opt.MAR$OTC$Stage1

## End(Not run)

# 4.2) Finding the optimal testing configuration using non-informative
# array testing without master pooling

# The following call reproduces results given by Hitt et al. (2018) for
# non-informative array testing without master pooling with an overall
# disease prevalence of p=0.01 and sensitivity and specificity equal
# to 0.90.

# This example takes approximately 7 minutes to run.

## Not run:
results2 <- OTC(algorithm="A2", p=0.01, Se=0.90, Sp=0.90, group.sz=3:40,
obj.fn=c("ET", "MAR"))
results2$opt.ET$OTC
results2$opt.ET$ET/results2$opt.ET$OTC$Array.sz
results2$opt.MAR$OTC
results2$opt.MAR$ET/results2$opt.MAR$OTC$Array.sz

## End(Not run)
```

`accuracy.dorf` *Accuracy measures for informative Dorfman testing*

Description

Calculate the accuracy measures for each individual in a pool used with informative Dorfman testing.

Usage

```
accuracy.dorf(p, se, sp)
```

Arguments

p	a vector of each individual's probability of infection.
se	the sensitivity of the diagnostic test.
sp	the specificity of the diagnostic test.

Details

This function calculates the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual belonging to a pool of size greater than or equal to one used with informative Dorfman testing. Calculations of these measures are done using the equations presented in McMahan et al. (2012).

Value

a list containing:

PSe	a vector containing each individual's pooling sensitivity.
PSp	a vector containing each individual's pooling specificity.
PPV	a vector containing each individual's pooling positive predictive value.
NPV	a vector containing each individual's pooling negative predictive value.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

http://chrisbilder.com/grouptesting/

Other Informative Dorfman functions: characteristics.pool(), inf.dorf.measures(), opt.info.dorf(), opt.pool.size(), pool.specific.dorf(), thresh.val.dorf()

Examples

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8135)
p.vec <- p.vec.func(p=0.02, alpha=1, grp.sz=10)
accuracy.dorf(p=p.vec[1:3], se=0.90, sp=0.90)
```

`Array.Measures` *Operating characteristics for array testing without master pooling*

Description

Calculate the expected number of tests and accuracy measures for each individual using array testing without master pooling

Usage

```
Array.Measures(p, se, sp)
```

Arguments

`p`	matrix of probabilities corresponding to each individual's risk of disease.
`se`	the sensitivity of the diagnostic test.
`sp`	the specificity of the diagnostic test.

Details

This function calculates the operating characteristics for array testing without master pooling. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

Value

A list containing:

`T`	the expected number of tests for the array.
`PSe`	a matrix containing each individual's pooling sensitivity, corresponding to the input matrix of individual probabilities.
`PSp`	a matrix containing each individual's pooling specificity, corresponding to the input matrix of individual probabilities.
`PPV`	a matrix containing each individual's pooling positive predictive value, corresponding to the input matrix of individual probabilities.
`NPV`	a matrix containing each individual's pooling negative predictive value, corresponding to the input matrix of individual probabilities.

Note

This function returns the pooling positive and negative predictive values for all individuals in the array even though these measures are diagnostic specific; i.e., PPV (NPV) should only be considered for those individuals who have tested positive (negative).

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Two-Dimensional Informative Array Testing." *Biometrics*, **68**(3), 793–804. ISSN 0006341X, doi:10.1111/j.15410420.2011.01726.x, https://pubmed.ncbi.nlm.nih.gov/22212007/.

See Also

MasterPool.Array.Measures for calculating operating characteristics under non-informative array testing with master pooling, hierarchical.desc2 for three-stage hierarchical and non-informative two-stage hierarchical testing, and inf.dorf.measures for informative two-stage hierarchical testing. See p.vec.func for generating a vector of individual risk probabilities for informative group testing and Informative.array.prob for arranging individual risk probabilities in a matrix for informative array testing.

http://chrisbilder.com/grouptesting/

Other Operating characteristic functions: MasterPool.Array.Measures(), hierarchical.desc2(), inf.dorf.measures()

Examples

```
# Calculate the operating characteristics for
#   non-informative array testing without master
#   pooling, with a 5x5 array and an overall disease
#   risk of p = 0.02.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
p.mat <- matrix(data=0.02, ncol=5, nrow=5)
Array.Measures(p=p.mat, se=0.95, sp=0.95)

# Calculate the operating characteristics for
#   informative array testing without master
#   pooling, with a 3x3 array and an overall disease
#   risk of p = 0.03 and alpha = 2.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8791)
```

```
p.vec <- p.vec.func(p=0.03, alpha=2, grp.sz=9)
p.mat <- Informative.array.prob(prob.vec=p.vec, nr=3,
nc=3, method="gd")
Array.Measures(p=p.mat, se=0.99, sp=0.99)
```

`beta.dist` *Expected value of order statistics from a beta distribution*

Description

Get the expected value of order statistics, E(p(i)), from a beta distribution by specifying an average probability and shape parameters for the beta distribution.

Usage

```
beta.dist(
  p = 0.05,
  alpha = 1,
  beta = NULL,
  grp.sz = 10,
  simul = FALSE,
  plot = FALSE,
  rel.tol = ifelse(a >= 1, .Machine$double.eps^0.25, .Machine$double.eps^0.1)
)
```

Arguments

`p`	average probability, $\frac{\alpha}{\alpha+\beta}$.
`alpha`	the alpha parameter for the beta distribution. The details of the specification of `alpha` are given under 'Details'.
`beta`	the beta parameter for the beta distribution, which is calculated from the average probability, `p`, if it is not specified. The details of the specification of `beta` are given under 'Details'.
`grp.sz`	the number of individuals in the group.
`simul`	a logical value indicating whether to use simulation. If simulation is used, the vector of probabilities is found by simulating 10,000 values from a beta distribution with the specified shape parameters. If simulation is not used, the vector of individual probabilities is found using integration.
`plot`	a logical value indicating whether to plot the distribution with p(i) marked.
`rel.tol`	relative tolerance used for integration.

Details

If alpha = 0, this function uses an extreme value distribution based on a Bernoulli(p) distribution to find the individual probabilities, p_i. If alpha is infinite, this function uses $p_i = p$ for all i.

If beta is not specified, it is calculated from the average probability p as $b = a * \frac{1}{p-1}$. If beta is specified, this function ignores p unless alpha is infinite.

Depending on the specified probability, alpha level, and overall group size, simulation may be necessary in order to generate the vector of individual probabilities. In this case, the user should specify simul=TRUE and set a seed in order to reproduce results. See Black et al. (2015) for additional details.

Value

A vector of ordered probabilities, p_i.

Author(s)

This function was originally written by Michael S. Black for Black et al. (2015). The function was obtained from http://chrisbilder.com/grouptesting/.

References

Black, M., Bilder, C., Tebbs, J. (2015). "Optimal retesting configurations for hierarchical group testing." *Journal of the Royal Statistical Society. Series C: Applied Statistics*, **64**(4), 693–710. ISSN 14679876, doi:10.1111/rssc.12097.

See Also

http://chrisbilder.com/grouptesting/

p.vec.func for generating a vector of individual risk probabilities for informative group testing (after checking whether simulation is needed) and Informative.array.prob for arranging a vector of individual risk probabilities in a matrix for informative array testing without master pooling.

Other Individual risk probability functions: Informative.array.prob(), p.vec.func()

Examples

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8791)
beta.dist(p=0.05, alpha=1, grp.sz=30)

# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(1002)
beta.dist(p=0.02, alpha=2, grp.sz=50, simul=TRUE)
```

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(5732)
beta.dist(alpha=2, beta=5, grp.sz=15, plot=TRUE)
```

---

bgtCI *Confidence Intervals for One Proportion in Binomial Group Testing*

Description

Calculates the point estimate, the exact Clopper-Pearson and Blaker CI, the Score test derived Wilson and Agresti-Coull CI, the asymptotic second-order corrected interval fo Cai and the Wald CI for a single binomial proportion estimated from a binomial group testing trial. Assumes equal group sizes, an assay method classifying a group as positive if at least one unit in the group is positive, individuals units randomly assigned to the groups.

Usage

```
bgtCI(n, s, y, conf.level = 0.95,
 alternative = "two.sided", method = "CP")
```

Arguments

n	integer, specifying the number of groups (i.e. assays i.e. observations)
s	integer, specifying the common size of groups i.e. the number of individual units in each group
y	integer, specifying the number of positive groups
conf.level	nominal confidence level of the interval
alternative	character string defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' gives the only an upper bound with confidence level=conf.level 'greater' gives the only a lower bound with confidence level=conf.level and 'two.sided' gives a two-sided confidence interval with confidence level=conf.level
method	character string defining the method for CI calculation, where: "CP" is Clopper-Pearson, an exact tail interval showing symmetric coverage probability (inversion of two one-sided tests), "Blaker" is the Blaker interval, an exact interval, inversion of one two.sided test, therefore defined only two.sided, but shorter than the two-sided Clopper-Pearson CI. Both guarantee to contain the true parameter with at least conf.level*100 percent probability, "AC" is the Agresti-Coull (generalized Agresti-Coull) interval, asymptotic method, "Score" is Wilson Score, asymptotic method derived from inversion of the Score test, "SOC" is the second order corrected interval, asymptotic method for one-sided problems (for details see Cai, 2005), and "Wald" the Wald interval, which cannot be recommended.

Details

This function allows the computation of confidence intervals for binomial group testing as described in Tebbs & Bilder (2004) and Schaarschmidt (2007). If an actual confidence level greater or equal to that specified in the conf.level argument shall always be guaranteed, the exact method of Clopper-Pearson (`method="CP"`) can be recommended for one-sided and the improved method of Blaker (2000) (`method="Blaker"`) can be recommended for two-sided hypotheses. If a mean confidence level close to that specified in the argument conf.level is required, but moderate violation of this level is acceptable, the Second-Order corrected (`method="SOC"`), Wilson Score (`method="Score"`) or Agresti-Coull (`method="AC"`) might be used (Brown, Cai, DasGupta, 2001; Cai 2005).

Value

A list containing:

`conf.int`	a confidence interval for the proportion
`estimate`	the point estimator of the proportion

Author(s)

Frank Schaarschmidt

References

Blaker H (2000). Confidence curves and improved exact confidence intervals for discrete distributions. *The Canadian Journal of Statistics 28 (4), 783-798.*

Brown LD, Cai TT, DasGupta A (2001). Interval estimation for a binomial proportion. *Statistical Science 16 (2), 101-133.*

Cai TT (2005). One-sided confidence intervals in discrete distributions. *Journal of Statistical Planning and Inference 131, 63-88.*

Schaarschmidt F (2007). Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. *Communications in Biometry and Crop Science 2 (1), 32-40.* http://agrobiol.sggw.waw.pl/cbcs/

Tebbs JM & Bilder CR (2004). Confidence interval procedures for the probability of disease transmission in multiple-vector-transfer designs. *Journal of Agricultural, Biological and Environmental Statistics, 9 (1), 75-90.*

See Also

`pooledBin` for asymptotic confidence intervals and `bgtvs` for an exact confidence interval when designs with different group sizes are used

`bgtTest`: for hypothesis tests in binomial group testing

Examples

```
# See the example in Tebbs and Bilder (2004)
# the two.sided 95-percent
# Clopper-Pearson as default method:
```

```
bgtCI(n=24,y=3,s=7)
bgtCI(n=24,y=3,s=7,conf.level=0.95,
 alternative="two.sided", method="CP")

# other methods:
# Blaker CI is exact but shorter
# than Clopper-Pearson, only two.sided

bgtCI(n=24,y=3,s=7, alternative="two.sided",
 method="Blaker")

# the asymptotic Wilson CI might even
# be shorter:

bgtCI(n=24,y=3,s=7, alternative="two.sided",
 method="Score")

# one-sided confidence intervals:

bgtCI(n=24,y=3,s=7, alternative="less", method="CP")

# Wilson Score interval is less conservative
bgtCI(n=24,y=3,s=7, alternative="less", method="Score")

# the second-order corrected CI is even shorter
# in this situation:
bgtCI(n=24,y=3,s=7, alternative="less", method="SOC")
```

`bgtPower` *Power to Reject a Hypothesis in Binomial Group Testing for One Proportion*

Description

Closed calculation of the Power to reject a hypothesis in a binomial group testing experiment using confidence intervals for decision. Closed calculation of bias of the point estimator for a given experimental design n, s and the true, unknown proportion.

Usage

```
bgtPower(n, s, delta, p.hyp, conf.level = 0.95,
 method = "CP", alternative = "two.sided")
```

Arguments

`n`	integer, giving the number of groups i.e. assays i.e. observations, a vector of integers is also allowed
`s`	integer, giving the common size of groups i.e. the number of individual units in each group, a vector of integers is also allowed

`delta`	absolute difference between the threshold and the true proportion which shall be detectable with specified power, a vector is also allowed
`p.hyp`	number between 0 and 1, specifying the threshold proportion in the hypotheses
`conf.level`	confidence level required for the decision on the hypotheses
`method`	character string, specifying the Confidence interval method (see `bgtCI`) to be used
`alternative`	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' calculates the probability that `p.hyp` is excluded by an upper confidence limit for a true proportion p.hyp-delta, 'greater' calculates the probability that `p.hyp` is excluded by a lower confidence limit for a true proportion p.hyp+delta, 'two.sided' calculates min(power(p.hyp-delta, p.hyp+delta)) for a two-sided CI, thus can result in much lower power.

Details

The power of a confidence interval here is defined as the probability that a confidence interval or limit excludes the threshold parameter (p.hyp) of the null hypothesis, as described in Schaarschmidt(2007). I.e., the null hypothesis H0: p >= p.hyp might be rejected, if an upper confidence limit for p does not contain p.hyp. Due to discreteness, the power does not increase monotone for increasing number of groups n or group size s, but exhibits local maxima and minima, depending on n,s, conf.level, p.hyp. The power can be identical for different methods, depending on the particular combination of n, s, p.hyp, conf.level. Note that coverage probability and power are not necessarily symmetric for upper and lower bounds of binomial CI, especially for Wald, Wilson Score and Agresti-Coull CI.

Additional to the power, bias of the point estimator is calculated according to Swallow (1985).

If vectors are specified for n, s, and (or) delta, a matrix will be constructed and power and bias are calculated for each line in this matrix.

Value

A matrix containing the columns

`ns`	a vector of total sample size n*s resulting from the latter
`n`	a vector of number of groups
`s`	a vector of group sizes
`delta`	a vector of delta
`power`	the power to reject the given null hypothesis, with the specified method and parameters of the first 4 columns
`bias`	the bias of the estimator for the specified n, s, and the true proportion

References

Schaarschmidt F (2007). Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. Communications in Biometry and Crop Science 2 (1), 32-40. http://agrobiol.sggw.waw.pl/cbcs/

Swallow WH (1985). Group testing for estimating infection rates and probabilities of disease transmission. Phytopathology Vol.75, N.8, 882-889.

See Also

nDesign: stepwise increasing n (for a fixed group size s) until a certain power is reached within a restriction of bias of the estimator sDesign: stepwise increasing s (for a fixed number of groups) until a certain power is reached within a restriction of bias of the estimator estDesign: selection of an appropriate design to achieve minimal mean square error of the estimator

Examples

```
# Calculate the power for the design
# in the example given in Tebbs and Bilder(2004):
# n=24 groups each containing 7 insects
# if the true proportion of virus vectors
# in the population would be 0.04 (4 percent),
# the power to reject H0: p>=0.1 using an
# upper Clopper-Pearson ("CP") confidence interval
# can be calculated using the following call:

bgtPower(n=24, s=7, delta=0.06, p.hyp=0.1,
 conf.level=0.95, alternative="less", method="CP")


# c(), seq() or rep() might be used to explore development
# of power and bias for varying n, s, delta. How much can
# we decrease the number of groups (costly assays to be performed)
# by pooling the same number of 320 individuals to groups of
# increasing size without largely decreasing power?

bgtPower(n=c(320,160,80,64,40,32,20,10,5),
 s=c(1,2,4,5,8,10,16,32,64),
 delta=0.01, p.hyp=0.02)


# How does power develop for increasing differences
# delta between the true proportion and the threshold proportion?

bgtPower(n=50, s=10, delta=seq(from=0, to=0.01, by=0.001),
 p.hyp=0.01, method="CP")


# use a more liberal method:

bgtPower(n=50, s=10, delta=seq(from=0, to=0.01, by=0.001),
 p.hyp=0.01, method="SOC")
```

---

bgtTest *Hypothesis Test for One Proportion in Binomial Group Testing*

Description

Calculates p values for hypotheses tests of binomial proportions estimated from binomial group testing experiments against a threshold proportion in the hypotheses. Exact test, Score test and Wald test are available methods. Assumes equal group sizes, 100 percent sensitivity and specificity of the assays to test the groups, and individuals units randomly assigned to the groups with identical true probability of success.

Usage

```
bgtTest(n, y, s, p.hyp, alternative = "two.sided",
 method = "Exact")
```

Arguments

n	integer, number of groups (i.e. assays i.e. observations)
y	integer, number of positive groups
s	integer, common size of groups i.e. the number of individual units in each group
p.hyp	number between 0 and 1, specifying the hypothetical threshold proportion to test against
alternative	character string defining the alternative hypothesis, either 'two.sided', 'less' or 'greater'
method	character string defining the test method to be used: can be one of 'Exact' for an exact test corresponding to the Clopper-Pearson confidence interval 'Score' for a Score test, corresponding to the Wilson confidence interval 'Wald' for a Wald test corresponding to the Wald confidence interval, not recommended

Value

A list containing:

p.value	the p value of the test
estimate	the estimated proportion
p.hyp	as input
alternative	as input
method	as input

References

Swallow WH (1985) Group testing for estimating infection rates and probabilities of disease transmission. *Phytopathology 75 (8), 882-889.*

Blyth, C and Still, H. (1983) Binomial confidence intervals. *Journal of the American Statistical Association 78, 108-116.*

Santner TJ and Duffy DE (1989) The statistical analysis of discrete data. Springer New York.

Remund KM, Dixon DA, Wright DL, Holden LR (2001) Statistical considerations on seed purity testing on transgenic traits. *Seed Science Research (11), 101-119.*

See Also

bgtCI for confidence intervals in binomial group testing

Examples

```
# Assume the experiment: Assays are performed on
# n=10 groups, each group is a bulk sample
# of s=100 individuals, aim is to show that
# less than 0.5 percent ('p<0.005') of the units
# of the population show a detrimental trait (positive assay).
# The assay is senstive to show a positive result if only 1
# unit in the bulk sample of 100 units is positive.
# y=1 positive assay and 9 negative assays are observed.

bgtTest(n=10,y=1,s=100,alternative="less",method="Exact",p.hyp=0.005)

# The exact test corresponds to the
# limits of the Clopper-Pearson confidence interval
# in the example of Tebbs and Bilder(2004):

bgtTest(n=24, y=3, s=7, alternative="two.sided",
 method="Exact", p.hyp=0.0543)

bgtTest(n=24, y=3, s=7, alternative="two.sided",
 method="Exact", p.hyp=0.0038)

# Further methods:

bgtTest(n=24, y=3, s=7, alternative="two.sided",
 method="Score", p.hyp=0.0516)

bgtTest(n=24, y=3, s=7, alternative="two.sided",
 method="Wald", p.hyp=0.0401)
```

---

bgtvs *Confidence Interval for One Proportion in Group Testing with Variable Group Sizes*

Description

Calculates confidence intervals for a single proportion in binomial group testing if groups of different size are evaluated

Usage

```
bgtvs(n, s, y, conf.level = 0.95, alternative = "two.sided",
 maxiter = 100)
```

Arguments

n	vector of integer values, specifying the number of groups of the different sizes
s	vector of integer values, specifying the group sizes, must be of same length as n
y	vector of integer values, specifying the number of positive groups among the n groups tested
conf.level	a single numeric value, the confidence level of the interval
alternative	a character string, with options "two.sided", "less", "greater"
maxiter	maximal number steps in iteration of confidence limits

Details

Hepworth (1996) describes methods for constructing confidence intervals in binomial group testing, if groups of different size are used. Currently, only the exact method (Hepworth, 1996, equation5, Table.5) is implemented. Note, that the algorithm becomes computationally very expensive if the number of different groups becomes larger than 3.

Value

A list containing

conf.int	a numeric vector, the lower and upper limits of the confidence interval
estimate	the point estimate
conf.level	as input
alternative	as imput
input	a matrix containing the input values of n (number of groups), s (group size), and y (number of positive pools)

moreover, some of the input arguments.

Author(s)

Frank Schaarschmidt

References

Hepworth, G (1996): Exact confidence intervals for proportions estimated by group testing. *Biometrics 52, 1134-1146.*

See Also

pooledBin for asymptotic methods to calculate confidence intervals for one proportion in designs with a number of different pool sizes. Note that pooledBin can handle larger number of different pool sizes than bgtvs

Examples

```
# Consider a very simple example,
# given in Hepworth (1996), table 5:
# 2 groups each containing 5 units,
# and 3 groups, each containing 2 units

# In the first setting (n=2, s=5) y=1 positive group
# has been observed, in the second setting (n=3, s=2),
# y=2 positive have been observed.

bgtvs(n=c(2,3), s=c(5,2), y=c(1,2))

##############################################

# Recalculate the example given in
# Hepworth (1996), table 5:

 bgtvs(n=c(2,3), s=c(5,2), y=c(0,0))
 bgtvs(n=c(2,3), s=c(5,2), y=c(0,1))
 bgtvs(n=c(2,3), s=c(5,2), y=c(0,2))
 bgtvs(n=c(2,3), s=c(5,2), y=c(0,3))
 bgtvs(n=c(2,3), s=c(5,2), y=c(1,0))
 bgtvs(n=c(2,3), s=c(5,2), y=c(1,1))
 bgtvs(n=c(2,3), s=c(5,2), y=c(1,2))
 bgtvs(n=c(2,3), s=c(5,2), y=c(1,3))
 bgtvs(n=c(2,3), s=c(5,2), y=c(2,0))
 bgtvs(n=c(2,3), s=c(5,2), y=c(2,1))
 bgtvs(n=c(2,3), s=c(5,2), y=c(2,2))
 bgtvs(n=c(2,3), s=c(5,2), y=c(2,3))
```

---

bgtWidth *Expected Width of Confidence Intervals in Binomial Group Testing*

Description

Calculation of expected value of the width of confidence intervals for one proportion in binomial group testing, in dependence of the number of groups, group size, confidence level and an assumed true proportion. Available for the confidence interval methods in bgtCI(binGroup).

Usage

```
bgtWidth(n, s, p, conf.level = 0.95, alternative = "two.sided",
 method = "CP")
```

Arguments

n	integer, giving the number of groups i.e. assays i.e. observations, vector of integers is also allowed

s	integer, giving the common size of groups i.e. the number of individual units in each group, vector of integers is also allowed
p	assumed true proportion of individuals showing the trait to be estimated, vector is also allowed
conf.level	required confidence level of the interval
alternative	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' calculates the expected width between the assumed true proportion p and the upper conf.level*100 percent-bound of a one-sided CI, 'greater' calculates the expected width between the true assumed proportion p and the lower conf.level*100 percent-bound of a one-sided CI, 'two.sided' calculates the expected width between the lower and the upper bound of a two-sided conf.level*100 percent-CI.
method	character string as in the method argumnet in bgtCI

Details

For calculation of expected interval width in the standard binomial estimation see, e.g., Brown et al. (2001). The calculation in case of binomial group testing is simply done by replacing the binomial probabilities by the probabilities P(Y=y) for group testing (see Tebbs and Bilder, 2004)

Value

A matrix containing the columns

ns	the resulting total number of units n*s
n	number of groups
s	group size
p	the assumed true proportion

and the calculated

expCIWidth	expected value of CI width as defined under argument alternative

Author(s)

Frank Schaarschmidt

References

Brown LD, Cai TT, DasGupta A (2001) Interval estimation for a binomial proportion. *Statistical Science 16 (2), 101-133.*

Schaarschmidt F (2007) Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. *Communications in Biometry and Crop Science 2 (1), 32-40.* http://agrobiol.sggw.waw.pl/cbcs/

Tebbs JM & Bilder CR (2004) Confidence interval procedures for the probability of disease transmission in multiple-vector-transfer designs. *Journal of Agricultural, Biological and Environmental Statistics 9 (1), 75-90.*

Examples

```
# There is a minimal expected CI length, if
# group size s is increased (fixed other parameters)
# the corresponding group size might be chosen:

bgtWidth(n=20, s=seq(from=1, to=200, by=10),
 p=0.01, alternative="less", method="CP" )

# and this depends largely on the assumed proportion p:

bgtWidth(n=20, s=seq(from=1, to=200, by=10),
 p=0.05, alternative="less", method="CP" )

bgtWidth(n=20, s=seq(from=1, to=200, by=10),
 p=0.005, alternative="less", method="CP" )
```

---

binCI *Confidence Intervals for One Binomial Proportion*

Description

Calculates the exact Clopper-Pearson and Blaker, the asymptotic second-order corrected, Wilson, Agresti-Coull and Wald confidence interval for a single binomial proportion

Usage

```
binCI(n, y, conf.level = 0.95, alternative = "two.sided",
 method = "CP")

binCP(n, y, conf.level=0.95, alternative="two.sided")
binBlaker(n,y,conf.level=0.95, tolerance=1e-04, alternative="two.sided")
binAC(n, y, conf.level=0.95, alternative="two.sided")
binSOC(n, y,conf.level=0.95,alternative="two.sided")
binWald(n, y, conf.level=0.95, alternative="two.sided")
binWilson(n, y,conf.level=0.95,alternative="two.sided")
```

Arguments

n	number of trials (number of individuals under observation)
y	number of successes (number of individuals showing the trait of interest)
conf.level	nominal confidence level
alternative	character string defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' gives the only an upper bound with confidence level=conf.level 'greater' gives the only a lower bound with confidence level=conf.level and 'two.sided' gives a two-sided confidence interval with confidence level=conf.level

method	character string defining the method for CI calculation: where "CP" is Clopper-Pearson, an exact tail interval showing symmetric coverage probability (inversion of two one-sided tests), "Blaker" is the Blaker interval, an exact interval, inversion of one two.sided test, therefore defined only two.sided, but shorter than the two-sided Clopper-Pearson CI. Both guarantee to contain the true parameter with at least conf.level*100 percent probability, "AC" is Agresti-Coull, generalized Agresti-Coull interval, asymptotic method, "Score" is Wilson Score, asymptotic method derived from inversion of the Score test, "SOC" is the second order corrected interval, asymptotic method for one-sided problems (for details see Cai, 2005), and "Wald" the Wald interval, which cannot be recommended.
tolerance	precision of computation for the bounds of the Blaker interval

Details

This function allows computation of confidence intervals for a binomial proportion from a standard binomial experiment. If an actual confidence level greater or equal to that specified in the conf.level argument shall always be guaranteed, the exact method of Clopper-Pearson (method="CP") can be recommended for one-sided and the improved method of Blaker (method="Blaker") can be recommended for two-sided hypotheses. If a mean confidence level close to that specified in the argument conf.level is required, but moderate violation of this level is acceptable, the Second-Order corrected (method="SOC"), Wilson Score (method="Wilson") or Agresti-Coull (method="AC") might be used, where SOC has the most symmetric coverage and Wilson and Agresti-Coull are in tendency conservative for the upper bound and proportions close to 0 and for the lower bound and proportions close to 1. The Wald CI might be used for large number of observations n>10000 or intermediate proportions.

For discussion of CI for a single binomial proportion see Brown et al. (2001) for two-sided and Cai (2005) for one-sided intervals.

Value

A list containing:

conf.int	the estimated confidence interval
estimate	the point estimator

And the method, conf.level and alternative specified in the function call.

Author(s)

Frank Schaarschmidt

References

Blaker H (2000) Confidence curves and improved exact confidence intervals for discrete distributions. *The Canadian Journal of Statistics 28 (4), 783-798.*

Brown LD, Cai TT, DasGupta A (2001) Interval estimation for a binomial proportion. *Statistical Science 16 (2), 101-133.*

Cai TT(2005) One-sided confidence intervals in discrete distributions. *Journal of Statistical Planning and Inference 131, 63-88.*

See Also

binom.test for the exact confidence interval and test, binTest to calculate p.values of the exact, Score and Wald test.

Examples

```
# Default method is the two-sided 95% Clopper-Pearson CI:

binCI(n=200, y=10)

# other methods might result in
# shorter intervals (but asymetric coverage):

binCI(n=200,y=10, method="Blaker")
binCI(n=200,y=10, method="Score")
```

---

binDesign *Sample Size Iteration for One Parameter Binomial Problem*

Description

This function increases the sample size until a maximal sample size or a prespecified power is achieved.

Usage

```
binDesign(nmax, delta, p.hyp, conf.level = 0.95,
 power = 0.8, method = "CP", alternative = "two.sided")
```

Arguments

nmax	integer, maximal number of trials (individuals under observation) allowed in the iteration
delta	absolute difference between the threshold and the true proportion which shall be detectable with the specified power
p.hyp	threshold proportion to test against in the hypothesis, specify as a value between 0 and 1
conf.level	Confidence level of the decision, default is 0.95
power	Level of power to be achieved to be specified as a probability between 0 and 1
method	character string specifying the CImethod to be used for evaluation, see argument method in bgtCI
alternative	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' calculates the probability that p.hyp is excluded by an upper confidence limit given that the true proportion is p.hyp-delta, 'greater' calculates the probability that p.hyp is excluded by a lower confidence limit given that the true proportion is p.hyp+delta. 'two.sided' calculates min(power(p.hyp-delta, p.hyp+delta)) for a two-sided CI, thus can result in much lower power.

Details

The power of a confidence interval here is defined as the probability that a confidence interval or limit excludes the threshold parameter (`p.hyp`) of the hypothesis.

This function increases the number of trials (number of individuals under observation) until a pre-specified power is reached. Since the power does not increase monotone with increasing n for binomial proportions but oscillates between local maxima and minima, the simple iteration given here will generally result in selecting those n, for which the given CI method shows a local minimum of coverage if the null hypothesis is true. The power can be identical for different methods, depending on the particular combination of n, p.hyp, conf.level.

Especially for large n, the calculation time may become large (particularly for Blaker). Then only the range of sample size which is of interest can be specified in nmax, f.e. as: nmax=c(150,300). Alternatively, the function binPower might be used instead to calculate power and bias only for some particular combinations of n, delta, p.hyp,... .

Note that coverage probability and power are not necessarily symmetric for upper and lower bound of binomial CI.

The results can be visualized by application of the function plot() to the returned object of this function.

Value

a list

`powerout`	power value, either the power of the first n exceeding the pre-specified power, or the maximal power achieved in the specified range of n if the specified power has not been reached
`nout`	corresponding sample size n (the number of trials) for which the prespecified power is reached, or the sample size n for which the maximal power has been reached within the specified range, if pre-specified power has not been reached.

and additional input and iteration values needed only for the function plot.binDesign.

Author(s)

Frank Schaarschmidt

References

Schaarschmidt, F. (2007). Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. *Communications in Biometry and Crop Science 2 (1), 32-40.* http://agrobiol.sggw.waw.pl/cbcs/

See Also

`binPower` for calculation of power, `plot.binDesign` for plot of the results

Examples

```
# Find a sample size for which the power to
# reject H0: p >= 0.1 in favor of HA: p < 0.1 is
# at least 0.9 (90 percent) in case that the
# true proportion is 0.04 (i.e. an absolute delta
# of 0.06 to the threshold proportion p.hyp=0.1)
# The exact one sided Clopper-Pearson CI shall be
# used with default confidence level = 0.95.


sasi<-binDesign( nmax=200, delta=0.06,
 p.hyp=0.1, alternative="less", method="CP", power=0.9)
sasi

#### One can also plot the result:

 plot(sasi)


# For larger sample sizes iteration can be very time consuming.
# Better to use only a smaller range of n then:

sasi<-binDesign( nmax=c(200,300), delta=0.03, p.hyp=0.1,
 alternative="less", method="CP", power=0.9)
sasi
```

`binPower` *Power Calculation for One Parameter Binomial Problem*

Description

Closed calculation of the power to reject a hypothesis using confidence intervals for a single binomial proportion, for specified sample size n, conf.level and an assumed absolute difference to the threshold parameter under the null hypothesis.

Usage

```
binPower(n, delta, p.hyp, conf.level = 0.95,
alternative = "two.sided", method = "CP")
```

Arguments

`n`	number of trials n in the binomial experiment, specify as a single integer
`delta`	assumed absolute difference of the true proportion to the threshold proportion under the null hypothesis
`p.hyp`	threshold proportion under the null hypothesis

conf.level	nominal confidence level of the interval
alternative	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' calculates the 'power of the upper confidence limit' for a true proportion p.hyp - delta, 'greater' calculates the 'power of the lower confidece limit' for a true proportion of p.hyp + delta. 'two.sided' calculates min(power(p.hyp - delta, p.hyp + delta)) for a two-sided CI, thus can result in much lower power.
method	Character string, specifying the confidence interval method (see binCI) to be used

Details

The power of a confidence interval here is defined as the probability that a confidence interval or limit excludes the threshold parameter (p.hyp) of the null hypothesis. E.g., the null hypothesis H0: p>= 0.005 can be rejected, if an upper confidence limit for p does not contain p.hyp=0.005. In case that a delta of 0.002 shall be detectable, this function calculates the probability, that an interval of a given method will exclude p.hyp=0.005 if the true proportion = 0.003. Due to discreteness, the power does not increase monotone for increasing sample size (number of trials or indivoiduals under observation) n, but exhibits local maxima and minima, depending on n, conf.level and p.hyp. The power can be identical for different methods, depending on the particular combination of n, p.hyp, conf.level. Note, that coverage probability and power are not necessarily symmetric for upper and lower bound of binomial CI, especially for Wald, Wilson Score and Agresti-Coull CI.

Value

A list containing

power	the power which is achieved for the specified parameters and method

Author(s)

Frank Schaarschmidt

See Also

binDesign for iteration of a sample size n for which a specified power is reached

Examples

```
# What is the probability to reject the null hypothesis
# H0: p >= 0.02 in order to show that the alternative
# hypothesis HA: p < 0.02 is very likely in the first
# example of if 200 seeds are taken from a seed lot and
# are checked for the proportion of defectives.
# Assume a true proportion under the alternative:
# p = 0.01, i.e. a absolute difference delta = 0.01
# to the threshold proportion p.hyp=0.02.
# The null hypothesis can be rejected if the threshold
# p.hyp=0.02 is excluded by an 95 percent upper bound of the
# Clopper-Pearson CI. What is the probability  of this event?
```

```
binPower(n=200, delta=0.01, p.hyp=0.02,
 alternative="less", method="CP")

# Assuming a lower true proportion (if one is satisfied
# also with the situation that we can only reject H0
# in case that the seed lot has a very high purity, e.g.
# only a proportion of 0.001 defectives )

binPower(n=200, delta=0.019, p.hyp=0.02,
 alternative="less", method="CP")

# Or use a higher sample size:

binPower(n=600, delta=0.01, p.hyp=0.02,
 alternative="less", method="CP")
```

`binTest` *Hypothesis tests for One Binomial Proportion*

Description

Calculates p-values for hypothesis tests of a single binomial proportion.

Usage

```
binTest(n, y, p.hyp, alternative = "two.sided",
 method = "Exact")
```

Arguments

`n`	single integer value, number of trials (number of individuals under observation)
`y`	single integer value, number of successes (number of individuals showing the trait of interest)
`p.hyp`	single numeric value between 0 and 1, specifying the hypothetical threshold proportion to test against
`alternative`	character string defining the alternative hypothesis, either 'two.sided', 'less' or 'greater'
`method`	character string defining the test method to be used: can be one of `"Exact"` for an exact test corresponding to the Clopper-Pearson confidence interval, uses `binom.test(stats)` `"Score"` for a Score test, corresponding to the Wilson confidence interval `"Wald"` for a Wald test corresponding to the Wald confidence interval

Value

A list containing:

p.value	the p value of the test
estimate	the estimated proportion
p.hyp	as input
alternative	as input
method	as input

Author(s)

Frank Schaarschmidt

References

Santner, T.J. and Duffy, D.E. (1989) The statistical analysis of discrete data. Springer Verlag New York Berlin Heidelberg. Chapter 2.1.

See Also

binom.test(stats) for the exact test and corresponding confindence interval

Examples

```
# 200 seeds are taken from a seed lot.
# 2 are found to be defective.
# H0: p >= 0.02  shall be rejected in favor of  HA: p < 0.02.
# The exact test shall be used for decision:

binTest(n=200, y=2, p.hyp=0.02, alternative="less", method="Exact" )
```

---

binWidth *Expected Confidence Interval Width for One Binomial Proportion*

Description

Calculation of expected value of the width of confidence intervals in a binomial experiment, in dependence of the number of trials (number of individuals under observation), confidence level and an assumed true proportion. Available for the confidence interval methods in binCI(binGroup).

Usage

```
binWidth(n, p, conf.level = 0.95,
 alternative = "two.sided", method = "CP")
```

Arguments

n	integer, giving the number of trials (i.e. number of individuals under observation)
p	assumed true proportion of individuals showing the trait to be estimated
conf.level	required confidence level of the interval
alternative	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' calculates the expected width between the assumed true proportion p and the upper conf.level*100 percent-bound of a one-sided CI, 'greater' calculates the expected width between the assumed true proportion p and the lower conf.level*100 percent-bound of a one-sided CI, 'two.sided' calculates the expected width between the lower and the upper bound of a two-sided conf.level*100 percent-CI.
method	character string defining the method for CI calculation: where "CP" is Clopper-Pearson, an exact tail interval showing symmetric coverage probability (inversion of two one-sided tests), "Blaker" is the Blaker interval, an exact interval, inversion of one two.sided test, therefore defined only two.sided, but shorter than the two-sided Clopper-Pearson CI. Both guarantee to contain the true parameter with at least conf.level*100 percent probability, "AC" is Agresti-Coull, generalized Agresti-Coull interval, asymptotic method, "Score" is Wilson Score, asymptotic method derived from inversion of the Score test, "SOC" is the second order corrected interval, asymptotic method for one-sided problems (for details see Cai, 2005), and "Wald" the simple Wald-type interval.

Details

For calculation of expected interval width in the standard binomial estimation see Brown et al. (2001).

Value

A list containing:

expCIWidth	the expected value of the width of the confidence interval for the specified arguments

and the alternative, p and n which are specified in the function call.

Author(s)

Frank Schaarschmidt

See Also

binDesign for experimental design for hypothesis testing

Examples

```
# methods differ slightly in length when sample sizes are large:

binWidth(n=200,p=0.02,alternative="two.sided",
 method="CP")$expCIWidth

binWidth(n=200,p=0.02,alternative="two.sided",
 method="Blaker")$expCIWidth

binWidth(n=200,p=0.02,alternative="two.sided",
 method="Score")$expCIWidth

# but do more for small sample sizes and intermediate p:

binWidth(n=20,p=0.2,alternative="two.sided",
 method="CP")$expCIWidth

binWidth(n=20,p=0.2,alternative="two.sided",
 method="Blaker")$expCIWidth

binWidth(n=20,p=0.2,alternative="two.sided",
 method="Score")$expCIWidth
```

---

characteristics.pool *Testing expenditure for informative Dorfman testing*

Description

Calculate the expectation and variation of the testing expenditure of a pool used with informative Dorfman testing.

Usage

```
characteristics.pool(p, se, sp)
```

Arguments

p	a vector of each individual's probability of infection.
se	the sensitivity of the diagnostic test.
sp	the specificity of the diagnostic test.

Details

This function calculates the expected value and variance of the testing expenditure of a pool of size greater than or equal to one used with informative Dorfman testing. Calculations of these measures are done using the equations presented in McMahan et al. (2012).

Value

a list containing:

e	the expected testing expenditure of the pool.
v	the variation of the testing expenditure of the pool.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

http://chrisbilder.com/grouptesting/

Other Informative Dorfman functions: accuracy.dorf(), inf.dorf.measures(), opt.info.dorf(), opt.pool.size(), pool.specific.dorf(), thresh.val.dorf()

Examples

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8135)
p.vec <- p.vec.func(p=0.02, alpha=1, grp.sz=10)
characteristics.pool(p=p.vec[1:3], se=0.90, sp=0.90)
```

---

estDesign *Sample Size Iteration Depending on Minimal MSE in One-Parameter Group Testing*

Description

Find the group size s for a fixed number of assays n and an assumed true proportion p.tr for which the mean squared error (mse) of the point estimator is minimal and bias is within a restriction. For experimental design in binomial group testing as recommended by Swallow (1985), if main objective is estimation.

Usage

```
estDesign(n, smax, p.tr, biasrest = 0.05)
```

Arguments

n	integer, fixed sample size (number of assays)
smax	integer, maximal group size allowed in planning of the design
p.tr	assumed true proportion of the 'positive' trait in the population to be tested, specify as a value between 0 and 1
biasrest	value between 0 and 1 specifying the absolute bias maximally allowed

Details

Swallow (1985) recommends to use the upper bound of the expected range of true proportion p.tr for optimization of tzhe design. For further details see the reference. Up to now, specify n<1020.

Value

the group size s, for which the mse of the estimator is minimal for the given n, p.tr or the group size s for which bias restriction biasrest is just not violated, and for this particular group size s: a list containing:

varp	the variance of the estimator
mse	the mean square error of the estimator
bias	the bias of the estimator
exp	the expected value of the estimator

Author(s)

Frank Schaarschmidt

References

Swallow WH (1985) Group testing for estimating infection rates and probabilities of disease transmission. Phytopathology Vol.75, N.8, 882-889.

See Also

nDesign, sDesign for choice of the binomial group testing design according to the power in a hypothesis test

Examples

```
### Compare table 1 in Swallow(1985),885:

estDesign(n=10, smax=100, p.tr=0.001)

estDesign(n=10, smax=100, p.tr=0.01)

estDesign(n=25, smax=100, p.tr=0.05)

estDesign(n=40, smax=100, p.tr=0.25)
```

```
estDesign(n=200, smax=100, p.tr=0.3)
```

`gt.control` *Auxiliary for Controlling Group Testing Regression*

Description

Auxiliary function to control fitting parameters of EM algorithm used internally in `gtreg.mp` and `EM.mp` or `gtreg`, `EM`, and `EM.ret`.

Usage

```
gt.control(tol = 0.0001, n.gibbs = 1000, n.burnin = 20,
 maxit = 500, trace = FALSE, time = TRUE)
```

Arguments

`tol`	convergence criterion
`n.gibbs`	the Gibbs sample size to be used in each E step for the EM algorithm (default is 1000), for matrix pooling
`n.burnin`	the number of samples in the burn-in period (default is 20), for matrix pooling
`maxit`	maximal number of iterations in the EM algorithm
`trace`	logical indicating if output should be printed for each iteration, defaults to `FALSE`
`time`	logical indicating if the length of time for the model fitting should be printed, defaults to `TRUE`

Value

A list with components named as the arguments

Examples

```
# The default settings:
gt.control()
```

gtreg *Fitting Group Testing Models*

Description

gtreg is a function to fit the group testing regression model specified through a symbolic description of the linear predictor and descriptions of the group testing setting.

Usage

```
gtreg(formula, data, groupn, retest = NULL, sens = 1,
 spec = 1, linkf = c("logit", "probit", "cloglog"),
 method = c("Vansteelandt", "Xie"), sens.ind = NULL,
 spec.ind = NULL, start = NULL, control = gt.control(...), ...)

gtreg.fit(Y, X, groupn, sens, spec, linkf, start=NULL)

EM(Y, X, groupn, sens, spec, linkf, start = NULL,
 control = gt.control())

EM.ret(Y, X, groupn, ret, sens, spec, linkf, sens.ind,
 spec.ind, start = NULL, control = gt.control())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which gtreg is called.
groupn	a vector, list or data frame of the group numbers that designates individuals to groups.
retest	a vector, list or data frame of individual retest results for Dorfman's retesting procedure. Default value is NULL for no retests. See 'Details' for how to code it.
sens	sensitivity of the test, set to be 1 by default.
spec	specificity of the test, set to be 1 by default.
sens.ind	sensitivity of the individual retests, set to be equal to sens if not specified otherwise.
spec.ind	specificity of the individual retests, set to be equal to spec if not specified otherwise.
linkf	a character string specifying one of the three link functions for a binomial model: "logit" (default) or "probit" or "cloglog".

method	The method to fit the model, must be one of "Vansteelandt" (default) or "Xie". The option "Vansteelandt" finds estimates by directly maximizing the likelihood function based on the group responses while the option "Xie" uses the EM algorithm to maximize the likelihood function in terms of the unobserved individual responses.
start	starting values for the parameters in the linear predictor.
control	a list of parameters for controlling the fitting process in method "Xie". See the documentation for gt.control for details.
Y	For gtreg.fit, EM and EM.ret: the vector of the group responses.
X	For gtreg.fit, EM and EM.ret: the design matrix of the covariates.
ret	For EM.ret: a vector containing individual retest results.
...	arguments to be passed by default to gt.control: see argument control

Details

A typical predictor has the form groupresp ~ covariates where response is the (numeric) group response vector and covariates is a series of terms which specifies a linear predictor for individual responses. Note that it is actually the unobserved individual responses, not the observed group responses, which are modeled by the covariates here. In groupresp, a 0 denotes a negative response and a 1 denotes a positive response, where the probability of an individual positive response is being modeled directly. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. The terms in the formula will be reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on; to avoid this pass a terms object as the formula.

A specification of the form first:second indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

Three workhorse functions gtreg.fit, EM and EM.ret, where the first corresponds to Vansteelandt's method and the last two corresponds to Xie's method, are called by gtreg to carry out the model fitting. The gtreg.fit function uses the optim function with default method "Nelder-Mead" to maximize the likelihood function of the observed group responses. If this optimization method produces a Hessian matrix of all zero elements, the "SANN" method in optim is employed to find the coefficients and Hessian matrix. For "SANN" method, the number of iterations in optim is set to be 10000.

The EM and EM.ret function apply Xie's EM algorithm to the likelihood function written in terms of the unobserved individual responses; the functions use glm.fit to update the parameter estimates within each M step. The EM function is used when there are no retests and EM.ret is used when individual retests are available. Thus, within retest, individual observations in observed positive groups are 0 (negative) or 1 (positive); the remaining individual observations are NAs meaning that no retest is performed for them. Retests cannot be used with Vansteelandt's method; a warning message will be given in this case, and the individual retests will be ignored in the model fitting. There could be slight differences in the estimates between the Vansteelandt's and Xie's methods (when retests are not available) due to different convergence criteria.

The data used here should be in the form of simple pooling - meaning that each individual appears in exactly one pool. When only the group responses are observed, the null degrees of freedom are the number of groups minus 1 and the residual degrees of freedom are the number of groups minus

the number of parameters. When individual retests are observed too, it is an open research question for what the degrees of freedom and the deviance for the null model should be; therefore the degrees of freedom and `null.deviance` will not be displayed.

For the background on the use of `optim`, see `help(optim)`.

Value

`gtreg` returns an object of class `"gt"`. See later in this section. The function `summary` (i.e., `summary.gt`) can be used to obtain or print a summary of the results. The group testing functions `predict` (i.e., `predict.gt`) and `residuals` (i.e., `residuals.gt`) can be used to extract various useful features of the value returned by `gtreg`. An object of class `"gt"` is a list containing at least the following components:

`coefficients`	a named vector of coefficients
`hessian`	estimated Hessian matrix of the negative log likelihood function, serves as an estimate of the information matrix
`residuals`	the response residuals, difference of the observed group responses and the fitted group responses.
`fitted.values`	the fitted mean values of group responses.
`deviance`	the deviance between the fitted model and the saturated model.
`aic`	Akaike's An Information Criterion, minus twice the maximized log-likelihood plus twice the number of coefficients
`null.deviance`	The deviance for the null model, comparable with `deviance`. The null model will include only the intercept if there is one in the model.
`counts`	For Vansteelandt's method: the number of iterations in `optim`; For Xie's method: the number of iterations in the EM algorithm.
`df.residual`	the residual degrees of freedom.
`df.null`	the residual degrees of freedom for the null model.
`z`	the vector of group responses.
`call`	the matched call.
`formula`	the formula supplied.
`terms`	the terms object used.
`method`	the method (`"Vansteelandt"` or `"Xie"`) used to fit the model.
`link`	the link function used in the model.

Author(s)

Boan Zhang

References

Xie, M. (2001), Regression analysis of group testing samples, *Statistics in Medicine, 20, 1957-1969.*

Vansteelandt, S., Goetghebeur, E., and Verstraeten, T. (2000), Regression models for disease prevalence with diagnostic tests on pools of serum samples, *Biometrics, 56, 1126-1133.*

See Also

summary.gt, predict.gt and residuals.gt for gt methods. gtreg.mp for the group testing regression model in the matrix pooling setting.

Examples

```
data(hivsurv)

fit1 <- gtreg(formula = groupres ~ AGE + EDUC., data = hivsurv,
           groupn = gnum, sens = 0.9, spec = 0.9, method = "Xie")
fit1

## --- Continuing the Example from  '?sim.gt':

set.seed(46)
gt.data <- sim.gt(par = c(-12, 0.2), sample.size = 700, group.size = 5)
fit2 <- gtreg(formula = gres ~ x, data = gt.data, groupn = groupn)
fit2

set.seed(21)
gt.data <- sim.gt(par = c(-12, 0.2), sample.size = 700, group.size = 6,
               sens = 0.95, spec = 0.95, sens.ind = 0.98, spec.ind = 0.98)
fit1 <- gtreg(formula = gres ~ x, data = gt.data, groupn = groupn,
        retest = retest, method = "X", sens = 0.95, spec = 0.95, sens.ind = 0.98,
        spec.ind = 0.98, trace = TRUE)
summary(fit1)
```

---

gtreg.halving *Fitting Group Testing Models Under the Halving Protocol*

Description

gtreg.halving is a function to fit the group testing regression model under the halving protocol specified through a symbolic description of the linear predictor and descriptions of the group testing setting.

Usage

```
gtreg.halving(formula, data, groupn, subg, retest, sens = 1,
 spec = 1, linkf = c("logit", "probit", "cloglog"),
 sens.ind = NULL, spec.ind = NULL, start = NULL, control = gt.control(...), ...)

EM.halving(Y, X, groupn, subg, ret, sens, spec, linkf, sens.ind,
 spec.ind, start = NULL, control = gt.control())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which gtreg.halving is called.
groupn	a vector, list or data frame of the group numbers that designates individuals to groups.
subg	a vector, list or data frame of the group numbers that designates individuals to subgroups.
retest	a vector, list or data frame of individual retest results within a subgroup that tests positive.
sens	sensitivity of the test, set to be 1 by default.
spec	specificity of the test, set to be 1 by default.
sens.ind	sensitivity of the individual retests, set to be equal to sens if not specified otherwise.
spec.ind	specificity of the individual retests, set to be equal to spec if not specified otherwise.
linkf	a character string specifying one of the three link functions for a binomial model: "logit" (default) or "probit" or "cloglog".
start	starting values for the parameters in the linear predictor.
control	a list of parameters for controlling the fitting process. See the documentation for gt.control for details.
Y	For EM.halving: the vector of the group responses.
X	For EM.halving: the design matrix of the covariates.
ret	For EM.halving: a vector containing individual retest results.
...	arguments to be passed by default to gt.control: see argument control

Details

A typical predictor has the form groupresp ~ covariates where response is the (numeric) group response vector and covariates is a series of terms which specifies a linear predictor for individual responses. Note that it is actually the unobserved individual responses, not the observed group responses, which are modeled by the covariates here. In groupresp, a 0 denotes a negative response and a 1 denotes a positive response, where the probability of an individual positive response is being modeled directly. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. The terms in the formula will be reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on; to avoid this pass a terms object as the formula.

A specification of the form first:second indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

The EM.halving function apply Xie's EM algorithm to the likelihood function written in terms of the unobserved individual responses; the functions use glm.fit to update the parameter estimates within each M step. In the halving protocol, if the initial group tests positive, it is split into two subgroups. The two subgroups are subsequently tested and if either subgroup tests positive, the third and final step is to test all individuals within the subgroup. Thus, within subg, subgroup responses in observed positive groups are 0 (negative) or 1 (positive); the remaining subgroup responses are NAs meaning that no tests are performed for them. The individual retests are similarly coded.

Value

gtreg.halving returns an object of class "gt". See later in this section. The function summary (i.e., summary.gt) can be used to obtain or print a summary of the results. The group testing functions predict (i.e., predict.gt) and residuals (i.e., residuals.gt) can be used to extract various useful features of the value returned by gtreg.halving. An object of class "gt" is a list containing at least the following components:

coefficients	a named vector of coefficients
hessian	estimated Hessian matrix of the negative log likelihood function, serves as an estimate of the information matrix
residuals	the response residuals, difference of the observed group responses and the fitted group responses.
fitted.values	the fitted mean values of group responses.
deviance	the deviance between the fitted model and the saturated model.
aic	Akaike's An Information Criterion, minus twice the maximized log-likelihood plus twice the number of coefficients
counts	the number of iterations in the EM algorithm.
z	the vector of group responses.
call	the matched call.
formula	the formula supplied.
terms	the terms object used.
method	the method "Xie" used to fit the model.
link	the link function used in the model.

Author(s)

Boan Zhang

References

Xie, M. (2001), Regression analysis of group testing samples, *Statistics in Medicine, 20, 1957-1969.*

See Also

summary.gt, predict.gt and residuals.gt for gt methods.

Examples

```
## --- Continuing the Example from  '?sim.halving':

set.seed(46)
gt.data <- sim.halving(par = c(-6, .1), gshape = 17, gscale = 1.4,
               sample.size = 5000, group.size = 5,
               sens = 0.95, spec = 0.95)
fit1 <- gtreg.halving(formula = gres ~ x, data = gt.data, groupn = groupn,
        start = c(-6, .1), subg = subgroup, retest = retest,
        sens = .95, spec = .95, trace = TRUE)
summary(fit1)
```

`gtreg.mp` *Fitting Group Testing Models in Matrix Pooling Setting*

Description

`gtreg.mp` is a function to fit the group testing regression model in the matrix pooling setting specified through a symbolic description of the linear predictor and descriptions of the group testing setting.

Usage

```
gtreg.mp(formula, data, coln, rown, arrayn, retest = NULL,
 sens = 1, spec = 1,  linkf = c("logit", "probit", "cloglog"),
 sens.ind = NULL, spec.ind = NULL,  start = NULL,
 control = gt.control(...), ...)

EM.mp(col.resp, row.resp, X, coln, rown, sqn, ret, sens, spec,
 linkf, sens.ind, spec.ind, start = NULL, control = gt.control())
```

Arguments

`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `gtreg.mp` is called.
`coln`	a vector, list or data frame that specifies column group number for each sample
`rown`	a vector, list or data frame that specifies row group number for each sample
`arrayn`	a vector, list or data frame that specifies array number for each sample

retest	a vector, list or data frame of individual retest results. A 0 denotes negative and 1 denotes positive. A NA denotes that no retest is performed for that individual. Default value is NULL for no retests.
sens	sensitivity of the group tests, set to be 1 by default.
spec	specificity of the group tests, set to be 1 by default.
sens.ind	sensitivity of the individual retests, set to be equal to sens if not specified otherwise.
spec.ind	specificity of the individual retests, set to be equal to spec if not specified otherwise.
linkf	a character string specifying one of the three link functions for a binomial model: "logit" (default) or "probit" or "cloglog".
start	starting values for the parameters in the linear predictor.
control	a list of parameters for controlling the fitting process. See the documentation for gt.control for details.
col.resp	For EM.mp: vector of group responses of column pools for all samples. 0 denotes negative and 1 denotes positive.
row.resp	For EM.mp: vector of group responses of row pools for all samples. 0 denotes negative and 1 denotes positive.
X	For EM.mp: the design matrix of the covariates.
sqn	For EM.mp: a vector that specifies array number
ret	For EM.mp: a vector containing individual retest results
...	arguments to be passed to gt.control: see argument control

Details

With matrix pooling, individual samples are placed in a matrix-like grid where samples are pooled within each row and within each column. This leads to two kinds of group responses: row and column group responses. Thus, a typical predictor has the form `cbind(col.resp, row.resp) ~ covariates` where col.resp is the (numeric) column group response vector and row.resp is the (numeric) row group response vector. The covariates term is a series of terms which specifies a linear predictor for individual responses. Note that it is actually the unobserved individual responses, not the observed group responses, which are modeled by the covariates. In col.resp and row.resp, a 0 denotes a negative response and 1 denotes a positive response, where the probability of an individual positive response is being modeled directly. A terms specification of the form `first + second` indicates all the terms in first together with all the terms in second with duplicates removed. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on; to avoid this pass a terms object as the formula.

A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification `first*second` indicates the cross of first and second. This is the same as `first + second + first:second`.

EM.mp is the workhorse function. It applies Xie's EM algorithm to the likelihood function written in terms of the unobserved individual responses. In each E step, the Gibbs sampling technique is used to estimate the conditional probabilities. Because of the large number of Gibbs samples needed

to achieve convergence, the model fitting process could be quite slow, especially when multiple positive rows and columns are observed. In this case, we can either increase the Gibbs sample size to help achieve convergence or loosen the convergence criteria by increasing `tol` at the expense of perhaps poorer estimates. If follow-up retests are performed, the retest results going into the model will help achieve convergence faster with the same Gibbs sample size and convergence criteria. In each M step, we use `glm.fit` to update the parameter estimates

Value

`gtreg.mp` returns an object of class `"gt.mp"` which inherits from the class `"gt"`. See later in this section. The function `summary` (i.e., `summary.gt.mp`) can be used to obtain or print a summary of the results. The group testing function `predict` (i.e., `predict.gt`) can be used to make predictions on `"gt.mp"` objects. An object of class `"gt.mp"` is a list containing at least the following components:

`coefficients`	a named vector of coefficients.
`hessian`	estimated Hessian matrix of the negative log likelihood function, serves as an estimate of the information matrix
`counts`	the number of iterations performed in the EM algorithm.
`Gibbs.sample.size`	the number of Gibbs samples generated in each E step.
`call`	the matched call.
`formula`	the formula supplied.
`terms`	the terms object used.
`link`	the link function used in the model.

Author(s)

Boan Zhang

References

Xie, M. (2001), Regression analysis of group testing samples, *Statistics in Medicine, 20, 1957-1969.*

See Also

`summary.gt.mp` and `predict.gt` for `gt.mp` methods. `gtreg` for the group testing regression model in the simple pooling setting.

Examples

```
## --- Continuing the Example from  '?sim.mp':
# 5*6 and 4*5 matrix
set.seed(9128)
sa1a<-sim.mp(par=c(-7,0.1), n.row=c(5,4), n.col=c(6,5),
 sens=0.95, spec=0.95)
sa1<-sa1a$dframe
```

```
## Not run:
fit1 <- gtreg.mp(formula = cbind(col.resp, row.resp) ~ x, data = sa1,
                 coln = coln, rown = rown, arrayn = arrayn,
                 sens = 0.95, spec = 0.95, tol = 0.005, n.gibbs = 2000, trace = TRUE)
fit1
summary(fit1)


## End(Not run)

## Here is an example of how long this fitting process may take. For the
## following simulated data, it takes a computer with 2.2GHZ processor and
## 3GB RAM about 6 minutes to achieve convergence.
set.seed(9012)
sa2a<-sim.mp(par=c(-7,0.1), n.row=c(10,10,10,10), n.col=c(10,10,10,10),
             sens=0.95, spec=0.95)
sa2<-sa2a$dframe

## Not run:
fit2 <- gtreg.mp(formula = cbind(col.resp, row.resp) ~ x, data = sa2,
                 coln = coln, rown = rown, arrayn = arrayn, retest = retest,
                 sens = 0.95, spec = 0.95, start = c(-7, 0.1), tol = 0.005)

fit2
summary(fit2)


## End(Not run)
```

`hierarchical.desc2` *Operating characteristics for hierarchical group testing*

Description

Calculate operating characteristics for hierarchical group testing with up to four stages, given a vector of individual probabilities and a testing configuration.

Usage

```
hierarchical.desc2(p, I2 = NULL, I3 = NULL, se = 1, sp = 1, order.p = TRUE)
```

Arguments

`p`	vector of probabilities corresponding to each individual's risk of disease.
`I2`	a numeric vector of pool sizes for stage 2 testing (used in hierarchical testing with at least three stages).

I3	a numeric vector of pool sizes for stage 3 testing (used in hierarchical testing with at least four stages).
se	the sensitivity of the diagnostic test.
sp	the specificity of the diagnostic test.
order.p	a logical value indicating whether the vector of individual probabilities needs to be sorted.

Details

This function calculates the operating characteristics for hierarchical group testing with up to four stages of testing. Operating characteristics calculated are expected number of tests, and pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value, for each individual and for the configuration overall.

If I2 is NULL, there are two stages of testing. If I3 is NULL but I2 has values, there are three stages of testing. If both I2 and I3 have values, there are four stages of testing.

Vectors I2 and I3 should be entered using notation that keeps track of all individuals through all stages (e.g. for a group of 10 individuals that splits into 5, 4, and 1 individual at stage 2, then into 3, 2, 2, 1, and 1 individual at stage 3 before individual testing at stage 4, then I2=c(5,4,1) and I3=c(3,2,2,1,1,1) so that the specimen that was tested individually at stage 2 is still numbered at stage 3 even though it will not be tested again).

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

ET	the expected number of tests.
stages	the number of stages in the testing algorithm.
group.size	the total number of individuals tested in the algorithm.
I2	pool sizes for the second stage of testing, or "individual testing" if there are only two stages of testing.
I3	pool sizes for the third stage of testing, or "individual testing" if there are only three stages of testing.
m1	the initial (stage 1) group size for two stage testing, or the number of subgroups originating from the initial group.
m2	the number of subgroups for each preceding group containing more than one individual, or "individual testing" if there are only two stages of testing.
m3	the number of subgroups for each preceding group containing more than one individual, or "individual testing" if there are only three stages of testing. NULL if there are only two stages of testing.

`individual.testerror`
: a data frame containing:

pse.vec a vector containing each individual's pooling sensitivity.

psp.vec a vector containing each individual's pooling specificity.

pppv.vec a vector containing each individual's pooling positive predictive value.

pnpv.vec a vector containing each individual's pooling negative predictive value.

`group.testerror`
: a vector containing:

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

`individual.probabilities`
: a vector containing each individual's probability of disease. If `order.p=TRUE`, this is the sorted vector of individual probabilities.

Note

This function returns the pooling positive and negative predictive values for all individuals even though these measures are diagnostic specific; i.e., PPPV (PNPV) should only be considered for those individuals who have tested positive (negative).

Author(s)

This function was originally written by Michael S. Black for Black et al. (2015). The function was obtained from http://chrisbilder.com/grouptesting/. Minor modifications were made to the function for inclusion in the binGroup package.

References

Black, M., Bilder, C., Tebbs, J. (2015). "Optimal retesting configurations for hierarchical group testing." *Journal of the Royal Statistical Society. Series C: Applied Statistics*, **64**(4), 693–710. ISSN 14679876, doi:10.1111/rssc.12097.

See Also

`Array.Measures` for calculating operating characteristics under array testing without master pooling, `MasterPool.Array.Measures` for non-informative array testing with master pooling, and `inf.dorf.measures` for informative two-stage hierarchical testing. See `p.vec.func` for generating a vector of individual risk probabilities for informative group testing.

http://chrisbilder.com/grouptesting/

Other Operating characteristic functions: `Array.Measures()`, `MasterPool.Array.Measures()`, `inf.dorf.measures()`

Examples

```
# Calculate the operating characteristics for
#   non-informative two-stage hierarchical testing
#   with an overall disease prevalence of p = 0.015
#   and an initial group size of 12.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
p.vec <- rep(x=0.015, times=12)
hierarchical.desc2(p=p.vec, I2=NULL, I3=NULL, se=0.95,
sp=0.95, order.p=FALSE)

# Calculate the operating characteristics for
#   non-informative three-stage hierarchical testing
#   with an overall disease prevalence of p = 0.04,
#   where an initial group of 20 individuals is
#   split into equally sized subgroups of 5 each.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
p.vec <- rep(x=0.04, times=20)
hierarchical.desc2(p=p.vec, I2=rep(x=5, times=4),
I3=NULL, se=0.99, sp=0.99, order.p=FALSE)

# Calculate the operating characteristics for
#   informative three-stage hierarchical testing
#   where an initial group of 10 individuals is
#   split into subsequent groups of 5, 4, and 1
#   individual.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.02 and a heterogeneity level
#   of alpha = 0.5. Depending on the specified probability,
#   alpha level, and overall group size, simulation may
#   be necessary in order to generate the vector of individual
#   probabilities. This is done using p.vec.func() and
#   requires the user to set a seed in order to reproduce
#   results.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(1002)
p.vec <- p.vec.func(p=0.02, alpha=0.5, grp.sz=10)
hierarchical.desc2(p=p.vec, I2=c(5,4,1), I3=NULL,
se=0.90, sp=0.90, order.p=TRUE)
```

`hivsurv` *Data from an HIV surveillance project*

Description

The hivsurv data set comes from an HIV surveillance project discussed in Verstraeten et al. (1998) and Vansteelandt et al. (2000). The purpose of the study was to estimate the HIV prevalence among pregnant Kenyan women in four rural locations of the country, using both individual and group testing responses. Blood tests were administered to each participating woman, and 4 covariates were obtained on each woman. Because the original group responses are unavailable, individuals are artificially put into groups of 5 here to form group responses. Only the 428 complete observations are given here.

Usage

```
data(hivsurv)
```

Format

A data frame with 428 observations on the following 8 variables.

DATE the date when each sample was collected

PAR. parity (number of children)

AGE age (in years)

MA.ST. marital status (1: single; 2: married (polygamous); 3: married (monogamous); 4: divorced; 5: widow)

EDUC. highest attained education level (1: no schooling; 2: primary school; 3: secondary school; 4: higher)

HIV individual response of HIV diagnosis (0: negative; 1: positive)

gnum the group number that designates individuals into groups

groupres the group response calculated from artificially formed groups

Source

Verstraeten, T., Farah, B., Duchateau, L., Matu, R. (1998), Pooling sera to reduce the cost of HIV surveillance: a feasibility study in a rural Kenyan district, *Tropical Medicine & International Health, 3, 747-750.*

Vansteelandt, S., Goetghebeur, E., and Verstraeten, T. (2000), Regression models for disease prevalence with diagnostic tests on pools of serum samples, *Biometrics, 56, 1126-1133.*

Examples

```
data(hivsurv)

str(hivsurv)
```

`Inf.Array` *Find the optimal testing configuration for informative array testing without master pooling*

Description

Find the optimal testing configuration (OTC) for informative array testing without master pooling and calculate the associated operating characteristics.

Usage

```
Inf.Array(p, Se, Sp, group.sz, obj.fn, weights = NULL, alpha = 2)
```

Arguments

`p`	the probability of disease, which can be specified as an overall probability of disease, from which a heterogeneous vector of individual probabilities will be generated, or a heterogeneous vector of individual probabilities specified by the user.
`Se`	the sensitivity of the diagnostic test.
`Sp`	the specificity of the diagnostic test.
`group.sz`	a single group size (representing the row/column size) for which to calculate the operating characteristics, or a range of group (row/column) sizes over which to find the OTC.
`obj.fn`	a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, `"ET"`, will always be calculated. Additional options include `"MAR"` (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and `"GR"` (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.
`weights`	a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.
`alpha`	a scale parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector. If a heterogeneous vector of individual probabilities is specified by the user, `alpha` can be specified as `NA` or will be ignored.

Details

This function finds the OTC and computes the associated operating characteristics for informative array testing without master pooling, implemented using the gradient arrangement described in McMahan et al. (2012). Array testing without master pooling involves amalgamating specimens in rows and columns for the first stage of testing. This function uses only square arrays, which is the

way array-based group testing is carried out in most real-world applications. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) or McMahan et al. (2012) at http://chrisbilder.com/grouptesting/ for additional details on the implementaion of informative array testing without master pooling.

The value(s) specified by `group.sz` represent the initial group (row/column) size. If a single value is provided for `group.sz`, operating characteristics will be calculated and no optimization will be performed. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`alpha` the level of heterogeneity used to generate the vector of individual probabilities.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Array.dim the row/column size for the first stage of testing.

Array.sz the overall array size (the square of the row/column size).

p.mat the sorted matrix of individual probabilities, arranged using the gradient method described by McMahan et al. (2012).

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

McMahan, C., Tebbs, J., Bilder, C. (2012). "Two-Dimensional Informative Array Testing." *Biometrics*, **68**(3), 793–804. ISSN 0006341X, doi:10.1111/j.15410420.2011.01726.x, https://pubmed.ncbi.nlm.nih.gov/22212007/.

See Also

NI.Array for non-informative array testing without master pooling, NI.A2M for non-informative array testing with master pooling, and OTC for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: Inf.D3(), Inf.Dorf(), NI.A2M(), NI.Array(), NI.D3(), NI.Dorf(), OTC()

Examples

```
# Find the OTC for informative array testing without
#   master pooling over a range of group (row/column) sizes.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.03 and a heterogeneity level
#   of alpha = 2. Depending on the specified probability,
#   alpha level, and overall group size, simulation may
#   be necessary in order to generate the vector of individual
#   probabilities. This is done using p.vec.func() and
#   requires the user to set a seed in order to reproduce
#   results.
# This examples takes approximately 30 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
set.seed(1002)
Inf.Array(p=0.03, Se=0.99, Sp=0.99, group.sz=3:20,
obj.fn=c("ET", "MAR"), alpha=2)
## End(Not run)

# This example takes less than 1 second to run.
```

```
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(1002)
Inf.Array(p=0.03, Se=0.99, Sp=0.99, group.sz=3:5,
obj.fn=c("ET", "MAR"), alpha=2)

# Find the OTC for a specified group (row/column) size
#   for informative array testing without master pooling.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(14849)
Inf.Array(p=p.vec.func(p=0.05, alpha=0.5, grp.sz=100),
Se=0.95, Sp=0.95, group.sz=10, obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1,1,10,10), nrow=2, ncol=2, byrow=TRUE),
alpha=NA)
```

`Inf.D3` *Find the optimal testing configuration for informative three-stage hierarchical testing*

Description

Find the optimal testing configuration (OTC) for informative three-stage hierarchical testing and calculate the associated operating charcteristics.

Usage

```
Inf.D3(p, Se, Sp, group.sz, obj.fn, weights = NULL, alpha = 2)
```

Arguments

`p`	the probability of disease, which can be specified as an overall probability of disease, from which a heterogeneous vector of individual probabilities will be generated, or a heterogeneous vector of individual probabilities specified by the user.
`Se`	the sensitivity of the diagnostic test.
`Sp`	the specificity of the diagnostic test.
`group.sz`	a single group size over which to find the OTC out of all possible testing configurations, or a range of group sizes over which to find the OTC.
`obj.fn`	a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, `"ET"`, will always be calculated. Additional options include `"MAR"` (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and `"GR"` (a linear combination of the expected number of tests, the number of

misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.

`weights` a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

`alpha` a scale parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector. If a heterogeneous vector of individual probabilities is specified by the user, `alpha` can be specified as `NA` or will be ignored.

Details

This function finds the OTC and computes the associated operating characteristics for informative three-stage hierarchical testing. This function finds the optimal testing configuration by considering all possible testing configurations. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) or Black et al. (2015) at http://chrisbilder.com/grouptesting/ for additional details on the implementation of informative three-stage hierarchical testing.

The value(s) specified by `group.sz` represent the initial (stage 1) group size. If a single value is provided for `group.sz`, the OTC will be found over all possible testing configurations for that initial group size. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`alpha` the level of heterogeneity used to generate the vector of individual probabilities.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`

a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Stage1 pool size for the first stage of testing, i.e. the initial group size.

Stage2 pool sizes for the second stage of testing.

p.vec the sorted vector of individual probabilities.

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Black, M., Bilder, C., Tebbs, J. (2015). "Optimal retesting configurations for hierarchical group testing." *Journal of the Royal Statistical Society. Series C: Applied Statistics*, **64**(4), 693–710. ISSN 14679876, doi:10.1111/rssc.12097.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

See Also

`NI.D3` for non-informative three-stage hierarchical testing and `OTC` for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: `Inf.Array()`, `Inf.Dorf()`, `NI.A2M()`, `NI.Array()`, `NI.D3()`, `NI.Dorf()`, `OTC()`

Examples

```
# Find the OTC for informative three-stage hierarchical
#   testing over a range of group sizes.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.05 and a heterogeneity level
#   of alpha = 0.5. Depending on the specified probability,
#   alpha level, and overall group size, simulation may
#   be necessary in order to generate the vector of individual
```

```
#   probabilities. This is done using p.vec.func() and
#   requires the user to set a seed in order to reproduce
#   results.
# This example takes approximately 20 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
set.seed(8318)
Inf.D3(p=0.05, Se=0.99, Sp=0.99, group.sz=3:30,
obj.fn=c("ET", "MAR"), alpha=0.5)
## End(Not run)

# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8318)
Inf.D3(p=0.05, Se=0.99, Sp=0.99, group.sz=10:15,
obj.fn=c("ET", "MAR"), alpha=0.5)

# Find the OTC out of all possible testing configurations
#   for a specified group size and vector of individual
#   probabilities.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(1216)
p.vec <- p.vec.func(p=0.10, alpha=2, grp.sz=12)
Inf.D3(p=p.vec, Se=0.99, Sp=0.99, group.sz=12,
obj.fn=c("ET", "MAR", "GR"), weights=matrix(data=c(1,1),
nrow=1, ncol=2, byrow=TRUE), alpha=NA)
```

`Inf.Dorf` *Find the optimal testing configuration for informative two-stage hierarchical (Dorfman) testing*

Description

Find the optimal testing configuration (OTC) for informative two-stage hierarchical (Dorfman) testing and calculate the associated operating characteristics.

Usage

```
Inf.Dorf(p, Se, Sp, group.sz, obj.fn, weights = NULL, alpha = 2)
```

Arguments

p	the probability of disease, which can be specified as an overall probability of disease, from which a heterogeneous vector of individual probabilities will be generated, or a heterogeneous vector of individual probabilities specified by the user.
Se	the sensitivity of the diagnostic test.
Sp	the specificity of the diagnostic test.
group.sz	a single block size for which to find the OTC out of all possible configurations, or a range of block sizes over which to find the OTC.
obj.fn	a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, "ET", will always be calculated. Additional options include "MAR" (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and "GR" (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.
weights	a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.
alpha	a scale parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector. If a heterogeneous vector of individual probabilities is specified by the user, alpha can be specified as NA or will be ignored.

Details

This function finds the OTC and computes the associated operating characteristics for informative two-stage hierarchical (Dorfman) testing, implemented via the pool-specific optimal Dorfman (PSOD) method described in McMahan et al. (2012). This function finds the optimal testing configuration by considering all possible testing configurations instead of using the greedy algorithm proposed for PSOD testing. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) or McMahan et al. (2012) at http://chrisbilder.com/grouptesting/ for additional details on the implementation of informative two-stage hierarchical (Dorfman) testing.

The value(s) specified by group.sz represent the overall block size used in the pool-specific optimal Dorfman (PSOD) method, where the overall group size is not tested. Instead, multiple initial pool sizes within this block are found and tested in the first stage of testing. The second stage of testing consists of individual retesting. For more details on informative two-stage hierarchical testing implemented via the PSOD method, see Hitt et al. (2018) and McMahan et al. (2012).

If a single value is provided for group.sz, the OTC will be found over all possible testing configurations. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm, or within the entire array

for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`alpha` the level of heterogeneity used to generate the vector of individual probabilities.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Block.sz the block size/overall group size, which is not tested.

pool.szs pool sizes for the first stage of testing.

p.vec the sorted vector of individual probabilities.

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Dorfman, R. (1943). "The Detection of Defective Members of Large Populations." *The Annals of Mathematical Statistics*, **14**(4), 436–440. ISSN 0003-4851, doi:10.1214/aoms/1177731363, https://www.jstor.org/stable/2235930.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

NI.Dorf for non-informative two-stage hierarchical (Dorfman) testing and OTC for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: Inf.Array(), Inf.D3(), NI.A2M(), NI.Array(), NI.D3(), NI.Dorf(), OTC()

Examples

```
# Find the OTC for informative two-stage hierarchical
#   (Dorfman) testing.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.01 and a heterogeneity level
#   of alpha = 2. Depending on the specified probability,
#   alpha level, and overall group size, simulation may
#   be necessary in order to generate the vector of individual
#   probabilities. This is done using p.vec.func() and
#   requires the user to set a seed in order to reproduce
#   results.
# This example takes approximately 20 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
set.seed(9245)
Inf.Dorf(p=0.01, Se=0.95, Sp=0.95, group.sz=3:30,
obj.fn=c("ET", "MAR"), alpha=2)
## End(Not run)

# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(9245)
Inf.Dorf(p=0.01, Se=0.95, Sp=0.95, group.sz=5:10,
obj.fn=c("ET", "MAR"), alpha=2)

# Find the OTC for informative two-stage hierarchical
#   (Dorfman) testing, for a specified block size.
# This example uses rbeta() to generate random probabilities
#   and requires the user to set a seed in order to reproduce
#   results.
```

```
# This example takes approximately 2.5 minutes to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
set.seed(8791)
Inf.Dorf(p=p.vec.func(p=0.03, alpha=0.5, grp.sz=50),
Se=0.90, Sp=0.90, group.sz=50, obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1,1,10,10), nrow=2, ncol=2, byrow=TRUE),
alpha=NA)
## End(Not run)
```

`inf.dorf.measures` *Operating characteristics for informative two-stage hierarchical (Dorfman) testing*

Description

Calculate the expected number of tests and accuracy measures for each individual using informative two-stage hierarchical (Dorfman) testing, given a vector of individual probabilities and a testing configuration.

Usage

```
inf.dorf.measures(prob, se, sp, N, pool.sizes)
```

Arguments

`prob`	vector of probabilities corresponding to each individual's risk of disease.
`se`	the sensitivity of the diagnostic test.
`sp`	the specificity of the diagnostic test.
`N`	block size/initial group size that is not tested. This is the total number of individuals being tested. The details of `block.sz` are given under 'Details'.
`pool.sizes`	a vector of pool sizes for the first stage of testing.

Details

This function utilizes the equations given by McMahan et al. (2012) for informative two-stage hierarchical (Dorfman) testing. It also repurposes functions written by Christopher S. McMahan (see http://chrisbilder.com/grouptesting/) for the implementation of informative Dorfman testing and directly uses functions written for the calculation of the associated operating characteristics. This function calculates the operating characteristics for informative two-stage hierarchical (Dorfman) testing. Operating characteristics calculated are expected number of tests, and pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

The specified `N` represents the block size used in the pool-specific optimal Dorfman (PSOD) method. This is the total number of individuals being tested by the algorithm. This block is not initially

tested. Instead, multiple initial pool sizes within this block are found and tested in the first stage of testing. The second stage of testing consists of individual retesting. For more information on block size specification, see McMahan et al. (2012).

Value

A list containing:

e	the expected number of tests needed to decode all N individuals.
v	the variance of the total number of tests needed to decode all N individuals.
summary	a matrix containing the pool, probability of disease, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual. The pool column identifies which pool each individual is contained in for the first stage of testing.

Note

This function returns the pooling positive and negative predictive values for all individuals even though these measures are diagnostic specific; i.e., PPPV (PNPV) should only be considered for those individuals who have tested positive (negative).

Author(s)

The majority of this function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/. Minor modifications were made to the function for inclusion in the binGroup package.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

Array.Measures for calculating operating characteristics under array testing without master pooling, MasterPool.Array.Measures for non-informative array testing with master pooling, and inf.dorf.measures for informative two-stage hierarchical testing. See p.vec.func for generating a vector of individual risk probabilities for informative group testing.

This function repurposed code from opt.info.dorf so that PSOD testing could be implemented using all possible testing configurations instead of a greedy algorithm. This function also used characteristics.pool and accuracy.dorf to calculate operating characteristics for the optimal set of pool sizes.

http://chrisbilder.com/grouptesting/

Other Operating characteristic functions: Array.Measures(), MasterPool.Array.Measures(), hierarchical.desc2()

Other Informative Dorfman functions: accuracy.dorf(), characteristics.pool(), opt.info.dorf(), opt.pool.size(), pool.specific.dorf(), thresh.val.dorf()

Examples

```
# Calculate the operating characteristics for
#   informative two-stage hierarchical (Dorfman) testing
#   with an overall disease prevalence of E(p(i)) = 0.01,
#   where a block size of 50 is split into initial pools
#   of 18, 13, 11, and 8 individuals.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8791)
inf.dorf.measures(prob=beta.dist(p=0.01, alpha=2,
grp.sz=50, simul=TRUE), se=0.95, sp=0.95, N=50,
pool.sizes=c(18, 13, 11, 8))
```

Informative.array.prob

Arrange a matrix of probabilities for informative array testing

---

Description

Arrange a vector of individual risk probabilities in a matrix for informative array testing without master pooling.

Usage

```
Informative.array.prob(prob.vec, nr, nc, method = "sd")
```

Arguments

prob.vec	vector of individual risk probabilities, of length nr*nc.
nr	the number of rows in the array.
nc	the number of columns in the array.
method	character string defining the method to be used for matrix arrangement. Options include spiral ("sd") and gradient ("gd") arrangement. See McMahan et al. (2012) for additional details.

Value

A matrix of probabilities arranged according to the specified method.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Two-Dimensional Informative Array Testing." *Biometrics*, **68**(3), 793–804. ISSN 0006341X, doi:10.1111/j.15410420.2011.01726.x, https://pubmed.ncbi.nlm.nih.gov/22212007/.

See Also

p.vec.func for generating a vector of individual risk probabilities from an overall probability of disease and Array.Measures for calculating operating characteristics for informative array testing without master pooling.

http://chrisbilder.com/grouptesting/

Other Individual risk probability functions: beta.dist(), p.vec.func()

Examples

```
# Use the gradient arrangement method to create a matrix
#   of individual risk probabilities for a 10x10 array.
# Depending on the specified probability, alpha level,
#   and overall group size, simulation may be necessary in
#   order to generate the vector of individual probabilities.
#   This is done using the p.vec.func() function and requires
#   the user to set a seed in order to reproduce results.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(1107)
p.vec1 <- p.vec.func(p=0.05, alpha=2, grp.sz=100)
Informative.array.prob(prob.vec=p.vec1, nr=10, nc=10, method="gd")

# Use the spiral arrangement method to create a matrix
#   of individual risk probabilities for a 5x5 array.
# Depending on the specified probability, alpha level,
#   and overall group size, simulation may be necessary in
#   order to generate the vector of individual probabilities.
#   This is done using the p.vec.func() function and requires
#   the user to set a seed in order to reproduce results.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8791)
p.vec2 <- p.vec.func(p=0.02, alpha=0.5, grp.sz=25)
Informative.array.prob(prob.vec=p.vec2, nr=5, nc=5, method="sd")
```

`MasterPool.Array.Measures` *Operating characteristics for array testing with master pooling*

Description

Calculate the expected number of tests and accuracy measures for each individual using array testing with master pooling.

Usage

```
MasterPool.Array.Measures(results, n, pmat, Se, Sp)
```

Arguments

`results`	an object containing results (expected number of tests and accuracy measures) from `Array.Measures`.
`n`	size of a row/column in the square array.
`pmat`	matrix of individual risk probabilities.
`Se`	the sensitivity of the diagnostic test.
`Sp`	the specificity of the diagnostic test.

Details

This function assumes that the array is square (i.e., the row and column size are equal) and utilizes the equations from Kim et al. (2007) for square array testing with master pooling. This function calculates the operating characteristics for array testing with master pooling. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

Value

A list containing:

`ET`	the expected number of tests for the array.
`PSe`	a matrix containing each individual's pooling sensitivity, corresponding to the input matrix of individual probabilities.
`PSp`	a matrix containing each individual's pooling specificity, corresponding to the input matrix of individual probabilities.
`PPV`	a matrix containing each individual's pooling positive predictive value, corresponding to the input matrix of individual probabilities.
`NPV`	a matrix containing each individual's pooling negative predictive value, corresponding to the input matrix of individual probabilities.

Note

This function returns the pooling positive and negative predictive values for all individuals in the array even though these measures are diagnostic specific; i.e., PPV (NPV) should only be considered for those individuals who have tested positive (negative).

Author(s)

Brianna D. Hitt

References

Kim, H., Hudgens, M., Dreyfuss, J., Westreich, D., Pilcher, C. (2007). "Comparison of group testing algorithms for case identification in the presence of test error." *Biometrics*, **63**(4), 1152–1163. ISSN 0006341X, doi:10.1111/j.15410420.2007.00817.x, https://pubmed.ncbi.nlm.nih.gov/17501946/.

See Also

`Array.Measures` for calculating operating characteristics under array testing without master pooling, `hierarchical.desc2` for three-stage hierarchical and non-informative two-stage hierarchical testing, and `inf.dorf.measures` for informative two-stage hierarchical testing.

Other Operating characteristic functions: `Array.Measures()`, `hierarchical.desc2()`, `inf.dorf.measures()`

Examples

```
# Calculate the operating characteristics for
#   non-informative array testing with master
#   pooling, with a 6x6 array and an overall
#   disease risk of p = 0.10.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
p.mat <- matrix(data=0.10, ncol=6, nrow=6)
results <- Array.Measures(p=p.mat, se=0.90, sp=0.90)
MasterPool.Array.Measures(results=results, n=36,
pmat=p.mat, Se=0.90, Sp=0.90)
```

---

nDesign *Iterate Sample Size in One Parameter Group Testing*

Description

Increasing number of groups (assays, bulk samples) for a fixed group size in a binomial group testing design, until a pre-specified power is achieved. At the same time, bias of the estimator is controlled. A hypothetical threshold proportion p.hyp and the absolute difference delta to be detected have to be specified.

Usage

```
nDesign(nmax, s, delta, p.hyp, conf.level = 0.95,
power = 0.8, alternative = "two.sided", method = "CP", biasrest = 0.05)
```

Arguments

nmax	either a single integer giving the maximal number of individuals n allowed in the iteration, or a vector of two integers giving the range of n in which power shall be iterated
s	integer, fixed group size (number of units per group)
delta	absolute difference between the threshold and the true proportion which shall be detectable with specified power
p.hyp	threshold proportion to test against in the hypothesis, specify as a value between 0 and 1
conf.level	confidence level of the decision, default is 0.95
power	level of power to be achieved to be specified as a probability between 0 and 1
alternative	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where 'less' calculates the probability that p.hyp is excluded by an upper confidence limit for a true proportion p.hyp - delta, 'greater' calculates the probability that p.hyp is excluded by a lower confidence limit for a true proportion p.hyp + delta. 'two.sided' calculates min(power(p.hyp - delta, p.hyp + delta)) for a two.sided CI, thus can result in much lower power.
method	character string specifying the CImethod to be used for evaluation, see argument method in bgtCI
biasrest	value between 0 and 1 specifying the absolute bias maximally allowed

Details

The power of a confidence interval here is defined as the probability that a confidence interval or limit excludes the threshold parameter (p.hyp) of the hypothesis.

This function increases the number of groups (i.e. number of observations or assays in binomial group testing) until a pre-specified power is reached or the maximal number of groups nmax (specified in the function call) is reached. Since the power does not increase monotone with increasing n for binomial proportions but oscillates between local maxima and minima, the simple iteration given here will generally result in selecting those n, for which the given CI method shows a local minimum of coverage if the null hypothesis is true. Bias decreases monotone with increasing the number of groups (if other parameters are fixed) The resulting Problems of chosing a number of groups which results in satisfactory power, are solved in the following manner:

In case that the pre-specified power can be reached within the given range of n, the smallest n will be returned for which at least this power is reached, as well as the actual power for this n.

In case that the pre-specified power is not reached within the given value, that n is returned for which maximal power is achieved, and the corresponding value of power.

In case that biasrestriction is violated even for the largest n within the given range of n, simply that n will be returned for which power was largest in the given range. Due to discreteness of binomial distribution, power can be zero for one-sided hypothesis over a range of n.

The power can be identical for different methods, depending on the particular combination of n, s, p.hyp, conf.level.

Especially for large n, the calculation time may become large (particularly for Blaker). Then only the range of sample size which is of interest can be specified in nmax, f.e. as: nmax=c(150,300). Alternatively, the function bgtPower might be used instead to calculate power and bias only for some particular combinations of n, s, delta, p.hyp,... .

Value

A list containing

`nout`	the number of groups (assays or bulk samples) necessary reach the power with the specified parameters
`powerout`	the power for the specified parameters and selected number of groups n
`biasout`	the bias for the specified parameters and the selected number of groups n

and a number of values specified in the function call or produced in the iteration, which are only necessary to apply the function plot() on objects of class 'nDesign'

Author(s)

Frank Schaarschmidt

References

Schaarschmidt F (2007). Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. Communications in Biometry and Crop Science 2 (1), 32-40. http://agrobiol.sggw.waw.pl/cbcs/

Swallow WH (1985). Group testing for estimating infection rates and probabilities of disease transmission. Phytopathology Vol.75, N.8, 882-889.

See Also

`plot.nDesign` to plot the iteration of this function

`bgtPower`: calculation of power and bias depending on n, s, delta, p.hyp, conf.level, method `sDesign`: function for stepwise increasing group size s for a given n in order to achieve sufficient power within a biasrestriction `estDesign`: function to choose group size s according to the minimal mse of the estimator, as given in Swallow (1985)

Examples

```
## Assume one aims to show that a proportion is smaller
## 0.005 (i.e. 0.5 per cent) with a power
## of 0.80 (i.e. 80 per cent) if the unknown proportion
## in the population is 0.003 (i.e. 0.3 per cent),
## thus, to detect a delta of 0.002.
## The Clopper Pearson CI shall be used.
## The maximal group size because of limited
## sensitivity of assay might be s=20 and we
```

```
## can only afford to perform maximally 100 assays:

nDesign(nmax=100, s=20, delta=0.002, p.hyp=0.005,
 alternative="less", method="CP", power=0.8)

## A power of 80 per cent can not be reached but
## only 30 percent with n=100
## One might accept to show significance only for a
## lower true proportion = 0.001 i.e accepting to be
## able to show significance only if true proportion
## is delta=0.004 smaller than the threshold

nDesign(nmax=100, s=20, delta=0.004, p.hyp=0.005,
 alternative="less", method="CP", power=0.8)


test<-nDesign(nmax=100, s=30, delta=0.004, p.hyp=0.005,
 alternative="less", method="CP", power=0.8)

plot(test)
```

NI.A2M *Find the optimal testing configuration for non-informative array testing with master pooling*

Description

Find the optimal testing configuration for non-informative array testing with master pooling and calculate the associated operating characteristics.

Usage

```
NI.A2M(p, Se, Sp, group.sz, obj.fn, weights = NULL)
```

Arguments

p	the probability of disease, which can be specified as an overall probability of disease or a homogeneous vector of individual probabilities.
Se	the sensitivity of the diagnostic test.
Sp	the specificity of the diagnostic test.
group.sz	a single group size (representing the row/column size) for which to calculate the operating characteristics, or a range of group (row/column) sizes over which to find the OTC.
obj.fn	a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, "ET", will always be calculated. Additional options include "MAR" (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)),

and "GR" (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.

`weights` a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

Details

This function finds the OTC and computes the associated operating characteristics for non-informative array testing with master pooling. Array testing with master pooling involves testing all specimens in the array together in one group before any row or column groups are formed. This function uses only square arrays, which is the way array-based group testing is carried out in most real-world applications. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ or Kim et al. (2007) for additional details on the implementation of non-informative array testing with master pooling.

The value(s) specified by `group.sz` represent the row/column size, which is used for the second stage of testing after the entire array is tested together in one group. If a single value is provided for `group.sz`, operating characteristics will be calculated and no optimization will be performed. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Array.dim the row/column size for the second stage of testing.

Array.sz the overall array size used for the first stage of testing.

p.mat the matrix of individual probabilities.

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Note

This function returns the pooling positive and negative predictive values for all individuals in the array even though these measures are diagnostic specific; i.e., PPV (NPV) should only be considered for those individuals who have tested positive (negative).

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Kim, H., Hudgens, M., Dreyfuss, J., Westreich, D., Pilcher, C. (2007). "Comparison of group testing algorithms for case identification in the presence of test error." *Biometrics*, **63**(4), 1152–1163. ISSN 0006341X, doi:10.1111/j.15410420.2007.00817.x, https://pubmed.ncbi.nlm.nih.gov/17501946/.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

See Also

NI.Array for non-informative array testing without master pooling, Inf.Array for informative array testing without master pooling, and OTC for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: Inf.Array(), Inf.D3(), Inf.Dorf(), NI.Array(), NI.D3(), NI.Dorf(), OTC()

Examples

```
# Find the OTC for non-informative array testing with
#   master pooling over a range of group (row/column) sizes.
# This example takes approximately 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
NI.A2M(p=0.04, Se=0.95, Sp=0.95, group.sz=3:10,
obj.fn=c("ET", "MAR"))

# Calculate the operating characteristics for a specified
#   group (row/column) size for non-informative array
#   testing with master pooling.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
NI.A2M(p=rep(0.01, 64), Se=0.90, Sp=0.90, group.sz=8,
obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1,1,10,10),
nrow=2, ncol=2, byrow=TRUE))
```

NI.Array *Find the optimal testing configuration for non-informative array testing without master pooling*

Description

Find the optimal testing configuration (OTC) for non-informative array testing without master pooling and calculate the associated operating characteristics.

Usage

```
NI.Array(p, Se, Sp, group.sz, obj.fn, weights = NULL)
```

Arguments

p	the probability of disease, which can be specified as an overall probability of disease or a homogeneous vector of individual probabilities.
Se	the sensitivity of the diagnostic test.
Sp	the specificity of the diagnostic test.
group.sz	a single group size (representing the row/column size) for which to calculate the operating characteristics, or a range of group (row/column) sizes over which to find the OTC. The details of group size specification are given under 'Details'.

`obj.fn` a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, `"ET"`, will always be calculated. Additional options include `"MAR"` (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and `"GR"` (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.

`weights` a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

Details

This function finds the OTC and computes the associated operating characteristics for non-informative array testing without master pooling. Array testing without master pooling involves amalgamating specimens in rows and columns for the first stage of testing. This function uses only square arrays, which is the way array-based group testing is carried out in most real-world applications. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ or Kim et al. (2007) for additional details on the implementation of non-informative array testing without master pooling.

The value(s) specified by `group.sz` represent the initial group (row/column) size. If a single value is provided for `group.sz`, operating characteristics will be calculated and no optimization will be performed. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Array.dim the row/column size for the first stage of testing.

Array.sz the overall array size (the square of the row/column size).

p.mat the matrix of individual probabilities.

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Kim, H., Hudgens, M., Dreyfuss, J., Westreich, D., Pilcher, C. (2007). "Comparison of group testing algorithms for case identification in the presence of test error." *Biometrics*, **63**(4), 1152–1163. ISSN 0006341X, doi:10.1111/j.15410420.2007.00817.x, https://pubmed.ncbi.nlm.nih.gov/17501946/.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

See Also

`Inf.Array` for informative array testing without master pooling, `NI.A2M` for non-informative array testing with master pooling, and `OTC` for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: `Inf.Array()`, `Inf.D3()`, `Inf.Dorf()`, `NI.A2M()`, `NI.D3()`, `NI.Dorf()`, `OTC()`

Examples

```
# Find the OTC for non-informative array testing
#   without master pooling over a range of group
#   (row/column) sizes.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
```

```
#   Intel i7-6500U processor.
NI.Array(p=0.04, Se=0.95, Sp=0.95, group.sz=3:10,
obj.fn=c("ET", "MAR"))

# Calculate the operating characteristics for a specified
#   group (row/column) size for non-informative array
#   testing without master pooling.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
NI.Array(p=rep(0.01, 64), Se=0.90, Sp=0.90, group.sz=8,
obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1,1,10,10,100,100),
nrow=3, ncol=2, byrow=TRUE))
```

`NI.D3` *Find the optimal testing configuration for non-informative three-stage hierarchical testing*

Description

Find the optimal testing configuration (OTC) for non-informative three-stage hierarchical testing and calculate the associated operating characteristics.

Usage

```
NI.D3(p, Se, Sp, group.sz, obj.fn, weights = NULL)
```

Arguments

`p`	the probability of disease, which can be specified as an overall probability of disease or a homogeneous vector of individual probabilities.
`Se`	the sensitivity of the diagnostic test.
`Sp`	the specificity of the diagnostic test.
`group.sz`	a single group size over which to find the OTC out of all possible testing configurations, or a range of group sizes over which to find the OTC.
`obj.fn`	a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, `"ET"`, will always be calculated. Additional options include `"MAR"` (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and `"GR"` (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.
`weights`	a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

Details

This function finds the OTC and computes the associated operating characteristics for non-informative three-stage hierarchical testing, by considering all possible testing configurations. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ or Kim et al. (2007) for additional details on the implementation of non-informative three-stage hierarchical testing.

The value(s) specified by `group.sz` represent the initial (stage 1) group size. If a single value is provided for `group.sz`, the OTC will be found over all possible testing configurations for that initial group size. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Stage1 pool size for the first stage of testing, i.e. the initial group size.

Stage2 pool sizes for the second stage of testing.

p.vec the vector of individual probabilities

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Kim, H., Hudgens, M., Dreyfuss, J., Westreich, D., Pilcher, C. (2007). "Comparison of group testing algorithms for case identification in the presence of test error." *Biometrics*, **63**(4), 1152–1163. ISSN 0006341X, doi:10.1111/j.15410420.2007.00817.x, https://pubmed.ncbi.nlm.nih.gov/17501946/.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

See Also

`Inf.D3` for informative three-stage hierarchical testing, `NI.Dorf` for non-informative two-stage hierarchical (Dorfman) testing, `Inf.Dorf` for informative two-stage hierarchical testing, and `OTC` for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: `Inf.Array()`, `Inf.D3()`, `Inf.Dorf()`, `NI.A2M()`, `NI.Array()`, `NI.Dorf()`, `OTC()`

Examples

```
# Find the OTC for non-informative three-stage
#   hierarchical testing over a range of group sizes.
# This example takes approximately 20 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
NI.D3(p=0.02, Se=0.90, Sp=0.90, group.sz=3:30,
obj.fn=c("ET", "MAR"))
## End(Not run)

# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
NI.D3(p=0.02, Se=0.90, Sp=0.90, group.sz=3:12,
obj.fn=c("ET", "MAR"))

# Find the OTC out of all possible configurations for
#   a specified group size for non-informative
#   three-stage hierarchical testing.
```

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
NI.D3(p=rep(0.005, 15), Se=0.99, Sp=0.99, group.sz=15,
obj.fn=c("ET", "MAR", "GR"), weights=matrix(data=c(1,1,10,10),
nrow=2, ncol=2, byrow=TRUE))
```

`NI.Dorf` *Find the optimal testing configuration for non-informative two-stage hierarchical testing*

Description

Find the optimal testing configuration (OTC) for non-informative two-stage hierarchical (Dorfman) testing and calculate the associated operating characteristics.

Usage

```
NI.Dorf(p, Se, Sp, group.sz, obj.fn, weights = NULL)
```

Arguments

`p`	the probability of disease, which can be specified as an overall probability of disease or a homogeneous vector of individual probabilities.
`Se`	the sensitivity of the diagnostic test.
`Sp`	the specificity of the diagnostic test.
`group.sz`	a single group size for which to calculate operating characteristics, or a range of group sizes over which to find the OTC. The details of group size specification are given under 'Details'.
`obj.fn`	a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, `"ET"`, will always be calculated. Additional options include `"MAR"` (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and `"GR"` (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.
`weights`	a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

Details

This function finds the OTC and computes the associated operating characteristics for non-informative two-stage hierarchical (Dorfman) testing. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for the algorithm. See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/, Dorfman (1943), or Kim et al. (2007) for additional details on the implementation of non-informative two-stage hierarchical testing.

The value(s) specified by `group.sz` represent the initial (stage 1) group size. If a single value is provided for `group.sz`, operating characteristics will be calculated and no optimization will be performed. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which include:

Stage1 pool size for the first stage of testing, i.e. the initial group size.

p.vec the vector of individual probabilities.

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Dorfman, R. (1943). "The Detection of Defective Members of Large Populations." *The Annals of Mathematical Statistics*, **14**(4), 436–440. ISSN 0003-4851, doi:10.1214/aoms/1177731363, https://www.jstor.org/stable/2235930.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Kim, H., Hudgens, M., Dreyfuss, J., Westreich, D., Pilcher, C. (2007). "Comparison of group testing algorithms for case identification in the presence of test error." *Biometrics*, **63**(4), 1152–1163. ISSN 0006341X, doi:10.1111/j.15410420.2007.00817.x, https://pubmed.ncbi.nlm.nih.gov/17501946/.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

See Also

Inf.Dorf for informative two-stage hierarchical (Dorfman) testing, NI.D3 for non-informative three-stage hierarchical testing, Inf.D3 for informative three-stage hierarchical testing, and OTC for finding the optimal testing configuration for a number of standard group testing algorithms.

http://chrisbilder.com/grouptesting/

Other OTC functions: Inf.Array(), Inf.D3(), Inf.Dorf(), NI.A2M(), NI.Array(), NI.D3(), OTC()

Examples

```
# Find the OTC for non-informative two-stage
#   hierarchical (Dorfman) testing over a range
#   of group sizes.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
NI.Dorf(p=0.01, Se=0.95, Sp=0.95, group.sz=2:100,
obj.fn=c("ET", "MAR"))

# Calculate the operating characteristics for a specified
#   initial group size for non-informative two-stage
#   hierarchical (Dorfman) testing.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
```

```
NI.Dorf(p=rep(0.025, 50), Se=0.90, Sp=0.90, group.sz=50,
obj.fn=c("ET", "MAR", "GR"), weights=matrix(data=c(1,1,10,10),
nrow=2, ncol=2, byrow=TRUE))
```

`opt.info.dorf` *Find the characteristics of an informative two-stage hierarchical (Dorfman) algorithm*

Description

Find the characteristics of an informative two-stage hierarchical (Dorfman) decoding process using Optimal Dorfman (OD), Thresholded Optimal Dorfman (TOD), or Pool-Specific Optimal Dorfman (PSOD) algorithms.

Usage

```
opt.info.dorf(
  prob,
  se = 1,
  sp = 1,
  method = "OD",
  max.pool = 15,
  thresh.pool = 8,
  threshold = NULL
)
```

Arguments

`prob`	a vector of all subjects' infection probabilities.
`se`	the sensitivity of the diagnostic test.
`sp`	the specificity of the diagnostic test.
`method`	character string defining the specific screening procedure for implementation of Dorfman retesting in a heterogeneous population. Options include Optimal Dorfman (`"OD"`), Thresholded Optimal Dorfman (`"TOD"`), and Pool-Specific Optimal Dorfman (`"PSOD"`). Further details are given under 'Details'.
`max.pool`	the maximum allowable pool size. Further details are given under 'Details'.
`thresh.pool`	the initial pool size used for TOD, if `threshold` is not specified. Further details are given under 'Details'.
`threshold`	the threshold value for TOD. If a threshold value is not specified, one is found algorithmically. Further details are given under 'Details'.

Details

This function finds the characteristics of an informative two-stage hierarchical (Dorfman) decoding process. Characteristics found include the expected expenditure of the decoding process, the variance of the expenditure of the decoding process, and the pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual. Calculations of these characteristics are done using equations presented in McMahan et al. (2012).

Optimal Dorfman (OD) is an informative Dorfman algorithm in which the common pool size $c = c_{opt}$ minimizes $E(T^(c))$, the expected number of tests needed to decode all N individuals when pools of size c are used.

Thresholded Optimal Dorfman (TOD) is an informative Dorfman algorithm in which all N individuals are partitioned into two classes, low-risk and high-risk individuals, based on whether their risk probability falls below or above a particular threshold value. The threshold can be specified using the `threshold` argument or the TOD algorithm can identify the optimal threshold value. The low-risk individuals are tested using a optimal common pool size, and high-risk individuals are tested individually.

Pool-Specific Optimal Dorfman (PSOD) is an informative Dorfman algorithm in which optimal sizes are determined for each pool. A total of N individuals are tested in pools that minimize the expected number of tests per individual, on a pool-by-pool basis. If desired, the user can add the constraint of a maximum allowable pool size, so that each pool will contain no more than the maximum allowable number of individuals.

All three informative Dorfman procedures described above require individuals to be ordered from smallest to largest probability of infection. See McMahan et al. (2012) for additional details on the implementation of informative two-stage hierarchical (Dorfman) testing algorithms.

Value

A list containing:

`tv`	the threshold value used for TOD, if applicable.
`e`	the expected expenditure of the decoding process.
`v`	the variance of the expenditure of the decoding process.
`summary`	a matrix of summary measures that includes each individual's infection probability, pool (pool to which they belong), pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

Dorfman, R. (1943). "The Detection of Defective Members of Large Populations." *The Annals of Mathematical Statistics*, **14**(4), 436–440. ISSN 0003-4851, doi:10.1214/aoms/1177731363, https://www.jstor.org/stable/2235930.

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

http://chrisbilder.com/grouptesting/

Other Informative Dorfman functions: accuracy.dorf(), characteristics.pool(), inf.dorf.measures(), opt.pool.size(), pool.specific.dorf(), thresh.val.dorf()

Examples

```
# Find the characteristics of an informative
#   Dorfman algorithm, using the OD procedure.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
opt.info.dorf(prob=rbeta(1000,1,10), se=1, sp=1,
method ="OD", max.pool=15, thresh.pool=8, threshold=NULL)

# Find the characteristics of an informative
#   Dorfman algorithm, using the TOD procedure.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(1002)
p.vec <- p.vec.func(p=0.01, alpha=2, grp.sz=20)
opt.info.dorf(prob=p.vec, se=0.95, sp=0.95,
method="TOD", max.pool=5, threshold=0.015)
```

---

opt.pool.size *Find the optimal pool size for Optimal Dorfman or Thresholded Optimal Dorfman*

Description

Find the optimal common pool size for Optimal Dorfman (OD) or Thresholded Optimal Dorfman (TOD) testing.

Usage

```
opt.pool.size(p, max.p, se = 1, sp = 1)
```

Arguments

p	a vector of each individual's probability of infection.
max.p	the maximum allowable pool size.
se	the sensitivity of the diagnostic test.
sp	the specificity of the diagnostic test.

Details

This function finds the optimal common pool size for OD or TOD testing. Using OD testing, all individuals are tested using an optimal common pool size. Using TOD testing, individuals are partitioned into low-risk and high-risk groups, and all low-risk individuals are tested using an optimal common pool size. See McMahan et al. (2012) for additional details on the implementation of OD or TOD testing.

Value

The optimal common pool size for OD or TOD testing.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

http://chrisbilder.com/grouptesting/

Other Informative Dorfman functions: accuracy.dorf(), characteristics.pool(), inf.dorf.measures(), opt.info.dorf(), pool.specific.dorf(), thresh.val.dorf()

Examples

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8135)
p.vec <- p.vec.func(p=0.02, alpha=1, grp.sz=10)
opt.pool.size(p=p.vec, max.p=3, se=0.95, sp=0.95)
```

OTC *Find the optimal testing configuration*

Description

Find the optimal testing configuration (OTC) for standard group testing algorithms and calculate the associated operating characteristics.

Usage

```
OTC(
  algorithm,
  p = NULL,
  probabilities = NULL,
  Se = 0.99,
  Sp = 0.99,
  group.sz,
  obj.fn = c("ET", "MAR"),
  weights = NULL,
  alpha = 2
)
```

Arguments

`algorithm`	character string defining the group testing algorithm to be used. Non-informative testing options include two-stage hierarchical (`"D2"`), three-stage hierarchical (`"D3"`), square array testing without master pooling (`"A2"`), and square array testing without master pooling (`"A2M"`). Informative testing options include two-stage hierarchical (`"ID2"`), three-stage hierarchical (`"ID3"`), and square array testing without master pooling (`"IA2"`).
`p`	overall probability of disease that will be used to generate a vector/matrix of individual probabilities. For non-informative algorithms, a homogeneous set of probabilities will be used. For informative algorithms, the `p.vec.func` function will be used to generate a heterogeneous set of probabilities. Either `p` or `probabilities` should be specified, but not both.
`probabilities`	a vector of individual probabilities, which is homogeneous for non-informative testing algorithms and heterogeneous for informative testing algorithms. Either `p` or `probabilities` should be specified, but not both.
`Se`	the sensitivity of the diagnostic test.
`Sp`	the specificity of the diagnostic test.
`group.sz`	a single group size or range of group sizes for which to calculate operating characteristics and/or find the OTC. The details of group size specification are given under 'Details'.

`obj.fn` a list of objective functions which are minimized to find the OTC. The expected number of tests per individual, `"ET"`, will always be calculated. Additional options include `"MAR"` (the expected number of tests divided by the expected number of correct classifications, described in Malinovsky et al. (2016)), and `"GR"` (a linear combination of the expected number of tests, the number of misclassified negatives, and the number of misclassified positives, described in Graff & Roeloffs (1972)). See Hitt et al. (2018) at http://chrisbilder.com/grouptesting/ for additional details.

`weights` a matrix of up to six sets of weights for the GR function. Each set of weights is specified by a row of the matrix.

`alpha` a shape parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector (for informative testing only).

Details

This function finds the OTC and computes the associated operating characteristics for standard group testing algorithms, as described in Hitt et al. (2018) at http://chrisbilder.com/grouptesting/.

Available algorithms include two- and three-stage hierarchical testing and array testing with and without master pooling. Both non-informative and informative group testing settings are allowed for each algorithm, except informative array testing with master pooling is unavailable because this method has not appeared in the group testing literature. Operating characteristics calculated are expected number of tests, pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value for each individual.

The value(s) specified by `group.sz` represent the initial (stage 1) group size for three-stage hierarchical testing and non-informative two-stage hierarchical testing. For informative two-stage hierarchical testing, the `group.sz` specified represents the block size used in the pool-specific optimal Dorfman (PSOD) method, where the initial group (block) is not tested. For more details on informative two-stage hierarchical testing implemented via the PSOD method, see Hitt et al. (2018) and McMahan et al. (2012a). For array testing without master pooling, the `group.sz` specified represents the row/column size for initial (stage 1) testing. For array testing with master pooling, the `group.sz` specified represents the row/column size for stage 2 testing. The group size for initial testing is overall array size, given by the square of the row/column size.

If a single value is provided for `group.sz` with array testing or non-informative two-stage hierarchical testing, operating characteristics will be calculated and no optimization will be performed. If a single value is provided for `group.sz` with three-stage hierarchical or informative two-stage hierarchical, the OTC will be found over all possible configurations. If a range of group sizes is specified, the OTC will be found over all group sizes.

The displayed pooling sensitivity, pooling specificity, pooling positive predictive value, and pooling negative predictive value are weighted averages of the corresponding individual accuracy measures for all individuals within the initial group for a hierarchical algorithm, or within the entire array for an array-based algorithm. Expressions for these averages are provided in the Supplementary Material for Hitt et al. (2018). These expressions are based on accuracy definitions given by Altman and Bland (1994a, 1994b).

Value

A list containing:

`prob` the probability of disease, as specified by the user.

`alpha` level of heterogeneity for the generated probability vector (for informative testing only).

`Se` the sensitivity of the diagnostic test.

`Sp` the specificity of the diagnostic test.

`opt.ET`, `opt.MAR`, `opt.GR`
a list for each objective function specified by the user, containing:

OTC a list specifying elements of the optimal testing configuration, which may include:

Stage1 pool size for the first stage of hierarchical testing, if applicable.

Stage2 pool sizes for the second stage of hierarchical testing, if applicable.

Block.sz the block size/initial group size for informative Dorfman testing, which is not tested.

pool.szs pool sizes for the first stage of testing for informative Dorfman testing.

Array.dim the row/column size for array testing.

Array.sz the overall array size for array testing (the square of the row/column size).

p.vec the sorted vector of individual probabilities, if applicable.

p.mat the sorted matrix of individual probabilities in gradient arrangement, if applicable.

ET the expected testing expenditure for the OTC.

value the value of the objective function per individual.

PSe the overall pooling sensitivity for the algorithm. Further details are given under 'Details'.

PSp the overall pooling specificity for the algorithm. Further details are given under 'Details'.

PPPV the overall pooling positive predictive value for the algorithm. Further details are given under 'Details'.

PNPV the overall pooling negative predictive value for the algorithm. Further details are given under 'Details'.

Author(s)

Brianna D. Hitt

References

Altman, D., Bland, J. (1994). "Diagnostic tests 1: sensitivity and specificity." *BMJ*, **308**, 1552.

Altman, D., Bland, J. (1994). "Diagnostic tests 2: predictive values." *BMJ*, **309**, 102.

Graff, L., Roeloffs, R. (1972). "Group testing in the presence of test error; an extension of the Dorfman procedure." *Technometrics*, **14**(1), 113–122. ISSN 15372723, doi:10.1080/00401706.1972.10488888, https://www.tandfonline.com/doi/abs/10.1080/00401706.1972.10488888.

Hitt, B., Bilder, C., Tebbs, J., McMahan, C. (2018). "The Optimal Group Size Controversy for Infectious Disease Testing: Much Ado About Nothing?!" Manuscript submitted for publication.

Malinovsky, Y., Albert, P., Roy, A. (2016). "Reader reaction: A note on the evaluation of group testing algorithms in the presence of misclassification." *Biometrics*, **72**(1), 299–302. ISSN 15410420, doi:10.1111/biom.12385.

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

McMahan, C., Tebbs, J., Bilder, C. (2012). "Two-Dimensional Informative Array Testing." *Biometrics*, **68**(3), 793–804. ISSN 0006341X, doi:10.1111/j.15410420.2011.01726.x, https://pubmed.ncbi.nlm.nih.gov/22212007/.

See Also

NI.Dorf for non-informative two-stage (Dorfman) testing, Inf.Dorf for informative two-stage (Dorfman) testing, NI.D3 for non-informative three-stage hierarchical testing, Inf.D3 for informative three-stage hierarchical testing, NI.Array for non-informative array testing, Inf.Array for informative array testing, and NI.A2M for non-informative array testing with master pooling.

http://chrisbilder.com/grouptesting/

Other OTC functions: Inf.Array(), Inf.D3(), Inf.Dorf(), NI.A2M(), NI.Array(), NI.D3(), NI.Dorf()

Examples

```
# Find the OTC for non-informative
#   two-stage hierarchical (Dorfman) testing
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
OTC(algorithm="D2", p=0.05, Se=0.99, Sp=0.99, group.sz=2:100,
obj.fn=c("ET", "MAR"))

# Find the OTC for informative
#   two-stage hierarchical (Dorfman) testing, implemented
#   via the pool-specific optimal Dorfman (PSOD) method
#   described in McMahan et al. (2012a), where the greedy
#   algorithm proposed for PSOD is replaced by considering
#   all possible testing configurations.
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.01 and a heterogeneity level
#   of alpha = 0.5. Depending on the specified probability,
#   alpha level, and overall group size, simulation may
#   be necessary in order to generate the vector of individual
#   probabilities. This is done using p.vec.func() and
#   requires the user to set a seed in order to reproduce
#   results.
# This example takes approximately 2.5 minutes to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
```

```
## Not run:
set.seed(52613)
OTC(algorithm="ID2", p=0.01, Se=0.95, Sp=0.95, group.sz=50,
obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1, 1, 10, 10, 0.5, 0.5),
nrow=3, ncol=2, byrow=TRUE), alpha=0.5)
## End(Not run)

# Find the OTC over all possible
#   testing configurations for a specified group size for
#   non-informative three-stage hierarchical testing
# This example takes approximately 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
OTC(algorithm="D3", p=0.001, Se=0.95, Sp=0.95, group.sz=18,
obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1, 1), nrow=1, ncol=2, byrow=TRUE))

# Find the OTC for non-informative
#   three-stage hierarchical testing
# This example takes approximately 20 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
OTC(algorithm="D3", p=0.06, Se=0.90, Sp=0.90,
group.sz=3:30, obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1, 1, 10, 10, 100, 100),
nrow=3, ncol=2, byrow=TRUE))
## End(Not run)

# Find the OTC over all possible configurations
#   for a specified group size, given a
#   heterogeneous vector of probabilities.
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
OTC(algorithm="ID3", probabilities=c(0.012, 0.014, 0.011,
0.012, 0.010, 0.015), Se=0.99, Sp=0.99, group.sz=6,
obj.fn=c("ET","MAR","GR"), weights=matrix(data=c(1, 1),
nrow=1, ncol=2, byrow=TRUE), alpha=0.5)

# Calculate the operating characteristics for a specified array size
#   for non-informative array testing without master pooling
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
OTC(algorithm="A2", p=0.005, Se=0.95, Sp=0.95, group.sz=8,
obj.fn=c("ET", "MAR"))
```

```
# Find the OTC for informative array testing without
#   master pooling
# A vector of individual probabilities is generated using
#   the expected value of order statistics from a beta
#   distribution with p = 0.03 and a heterogeneity level
#   of alpha = 2. The probabilities are then arranged in
#   a matrix using the gradient method described in
#   McMahan et al. (2012b). Depending on the specified
#   probability, alpha level, and overall group size,
#   simulation may be necessary in order to generate the
#   vector of individual probabilities. This is done using
#   p.vec.func() and requires the user to set a
#   seed in order to reproduce results.
# This example takes approximately 30 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
set.seed(1002)
OTC(algorithm="IA2", p=0.03, Se=0.95, Sp=0.95,
group.sz=3:20, obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1, 1, 10, 10, 100, 100),
nrow=3, ncol=2, byrow=TRUE), alpha=2)
## End(Not run)

# Find the OTC for non-informative array testing
#   with master pooling
# This example takes approximately 20 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
OTC(algorithm="A2M", p=0.02, Se=0.90, Sp=0.90,
group.sz=3:20, obj.fn=c("ET", "MAR", "GR"),
weights=matrix(data=c(1, 1, 10, 10, 0.5, 0.5, 2, 2,
100, 100, 10, 100), nrow=6, ncol=2, byrow=TRUE))
## End(Not run)
```

`p.vec.func` *Generate a vector of probabilities for informative group testing algorithms.*

Description

Generate a vector of individual risk probabilities using an overall probability of disease (i.e., the expected value of order statistics from a beta distribution) for use with informative group testing algorithms.

Usage

```
p.vec.func(p, alpha, grp.sz)
```

Arguments

p	overall probability of disease that will be used to generate a vector of individual risk probabilities.
alpha	a shape parameter for the beta distribution that specifies the degree of heterogeneity for the generated probability vector.
grp.sz	the number of total individuals for which to generate risk probabilities.

Details

This function uses Michael Black's beta.dist function to generate a vector of individual risk probabilities, ordered from least to greatest. Depending on the specified probability, alpha level, and overall group size, simulation may be necessary in order to generate the vector of individual probabilities. For this reason, the user should set a seed in order to reproduce results. The p.vec.func function augments the beta.dist function by checking whether simulation is needed before attempting to generate the vector of individual risk probabilities. See Black et al. (2015) for additional details on Michael Black's beta.dist function.

Value

A vector of individual risk probabilities.

Author(s)

Brianna D. Hitt

References

Black, M., Bilder, C., Tebbs, J. (2015). "Optimal retesting configurations for hierarchical group testing." *Journal of the Royal Statistical Society. Series C: Applied Statistics*, **64**(4), 693–710. ISSN 14679876, doi:10.1111/rssc.12097.

See Also

Informative.array.prob for arranging a vector of individual risk probabilities in a matrix for informative array testing without master pooling and beta.dist for the function on which p.vec.func is based on.

http://chrisbilder.com/grouptesting/

Other Individual risk probability functions: Informative.array.prob(), beta.dist()

Examples

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8791)
p.vec.func(p=0.03, alpha=0.5, grp.sz=100)

# This example takes less than 1 second to run.
```

```
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(52613)
p.vec.func(p=0.005, alpha=2, grp.sz=40)
```

`plot.bgtDesign` *Plot Results of nDesign or sDesign*

Description

Plots the results of the iteration performed by `nDesign` or `sDesign` in order to find an experimental design with sufficient power.

Usage

```
## S3 method for class 'nDesign'
plot(x, ...)
## S3 method for class 'sDesign'
plot(x, ...)
```

Arguments

`x`	An object of class `"nDesign"` or `"sDesign"` as can be created by `nDesign` and `sDesign`, respectively.
`...`	further arguments to be passed to `plot`

Examples

```
plot(nDesign(nmax=100, s=30, delta=0.004, p.hyp=0.005,
 alternative="less", method="Score", power=0.8))
```

`plot.binDesign` *Plot Results of binDesign*

Description

Plot function to visualize the power curve.

Usage

```
## S3 method for class 'binDesign'
plot(x, ...)
```

Arguments

x	an object of class "binDesign, as can be created by link{binDesign}"
...	plot parameters as described in par

Value

A plot.

See Also

binPower for calculation of power

Examples

```
# Find a sample size for which the power to reject
# H0: p >= 0.1 in favor of HA: p < 0.1
# is at least 0.9 (90 percent) in case that
# the true proportion is 0.04 (i.e. an absolute delta
# of 0.06 to the threshold proportion p.hyp=0.1)
# The exact one sided Clopper-Pearson CI shall be used
#  with default confidence level = 0.95.


sasi<-binDesign( nmax=200, delta=0.06, p.hyp=0.1,
 alternative="less", method="CP", power=0.9)

sasi

# Plot the result

# plot(sasi)


# for larger sample sizes this can be very time consuming.
# Better to use only a smaller range of n then:

sasi<-binDesign( nmax=c(200,300), delta=0.03, p.hyp=0.1,
 alternative="less", method="CP", power=0.9)

plot(sasi)
```

---

plot.poolbin *Diagnostic line fit for pool.bin objects*

Description

Diagnostic line fit for pool.bin objects

Usage

```
## S3 method for class 'poolbin'
plot(x, pch = 16, refline = TRUE, printR2 = TRUE, ...)
```

Arguments

x	an object of class "poolbin", as can be created by pooledBin
pch	symbol type for plotting, see ?pch
refline	logical, whether or not to add a reference line to the plot
printR2	logical, whether to print R^2
...	further arguments passed to plot

Details

Brad: Maybe drop a line here...

Value

a plot

Author(s)

Brad Biggerstaff

References

Brad: Any reference for this?

Examples

```
test <- pooledBin(x=c(8,3,1,0), m=c(50,20,10,5),
n=c(10,10,10,10), pt.method="firth", ci.method="lrt")

plot(test)
```

---

pool.specific.dorf *Find the optimal pool sizes for Pool-Specific Optimal Dorfman (PSOD) testing*

Description

Find the set of optimal pool sizes for Pool-Specific Optimal Dorfman (PSOD) testing.

Usage

```
pool.specific.dorf(p, max.p, se, sp)
```

Arguments

p	a vector of each individual's probability of infection.
max.p	the maximum allowable pool size.
se	the sensitivity of the diagnostic test.
sp	the specificity of the diagnostic test.

Details

This function finds the set of optimal pool sizes for PSOD testing. PSOD testing uses a greedy algorithm and does not consider all possible sets of pool sizes. See McMahan et al. (2012) for additional details on the implementation of PSOD testing.

Value

The optimal set of pool sizes for PSOD testing.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

http://chrisbilder.com/grouptesting/

Other Informative Dorfman functions: accuracy.dorf(), characteristics.pool(), inf.dorf.measures(), opt.info.dorf(), opt.pool.size(), thresh.val.dorf()

Examples

```
# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
set.seed(8135)
p.vec <- p.vec.func(p=0.02, alpha=1, grp.sz=10)
pool.specific.dorf(p=p.vec, max.p=3, se=0.95, sp=0.95)
```

pooledBin *Confidence intervals for a single proportion*

Description

Calculates confidence intervals for a single proportion based on pooled testing experiments containing various different pool sizes.

Usage

```
pooledBin(x, m, n = rep(1, length(x)),
 pt.method = c("firth", "gart", "bc-mle", "mle", "mir"),
 ci.method = c("skew-score", "bc-skew-score", "score",
 "lrt", "wald", "mir"),
 scale = 1, alpha = 0.05, tol = .Machine$double.eps^0.5)
```

Arguments

x	a vector, specifying the observed number of positive pools, among the number of pools tested (n)
m	a vector of pool sizes, must have the same length as, x
n	a vector of the corresponding number of pools of sizes m
pt.method	a character string, specifying the point estimate to compute, with the following options: "bc-mle" bias-corrected MLE, the default; "mle" MLE, and "mir" MIR.
ci.method	a character string, specifying the confidence interval to compute, with options: "skew-score" skewness-corrected, the default, "score" the score, "bc-skew-score" bias- and skewness-corrected "lrt" likelihood ratio test, "wald" wald, and "mir" MIR.
scale	a single numeric, coefficient to scale the point estimates and intervals bounds in the print and summary method (print.poolbin, summary.poolbin)
alpha	a single numeric, specifying the type-I-error level
tol	accuracy required for iterations in internal functions

Details

Point estimation: bias preventative ("firth") and bias corrected ("gart") estimators are recommended, with details described in Hepworth G, Biggerstaff BJ (2017). Use of MLE ("mle"), and MIR ("mir") estimators is not recommended.

Confidence intervals: Note, that the methods "mir" and "wald" can not be recommended, because they return too narrow intervals in relevant situations, "mir" because it ignores the pooling, and "wald" because it relies on simple large sample methods. For computational details and simulation results of the remaining methods, see Biggerstaff (2008).

Value

A list with elements

p	the estimated proportion
lcl	the lower confidence limit
ucl	the upper confidence limit
pt.method	the method used for point estimation
ci.method	the method used for interval estimation
alpha	the type-I-error level
x	the numbers of postive pools
m	the size of the pools
n	the numbers of pools with corresponding pool sizes m
scale	Scaling coefficient for the output

Author(s)

Brad Biggerstaff

References

Walter SD, Hildreth SW, Beaty BJ: Estimation of infection rates in population of organisms using pools of variable size. *Am J Epidemiol 1980, 112(1):124-128*

Hepworth G: Estimation of proportions by group testing. PhD Dissertation. Melbourne, Australia: The University of Melbourne; 1999.

Biggerstaff BJ (2008): Confidence interval for the difference of proportions estmimated from pooled samples. *Journal of Agricultural Biological and Environmental Statistics 2008, 13(4):478-496.*

Hepworth G, Biggerstaff BJ: Bias correction in estimating Proportions by pooled testing. *JABES 20178, to appear.*

See Also

bgtvs to compute exact confidence intervals for one proportion in designs with different pool sizes. Note that bgtvs can only deal with a limited number of different pool sizes.

bgtCI to compute exact or asymptotic confidence intervals for one proportion in designs with a common pool size in all pools.

Examples

```
# Consider an imaginary example, where pools of size
# 1, 5, 10 and 50 are tested, 5 pools of each size
# among the 5 pools with size 1 and 5, no pool is positive,
# while among the 5 pools of size 10 and 50, 1 and 2 positive
# pools are identified, respectively.

x1 <- c(0,0,1,2)
m1 <- c(1,5,10,50)
```

```
n1 <- c(5,5,5,5)

pooledBin(x=x1, m=m1, n=n1)
pooledBin(x=x1, m=m1, n=n1, scale=1000)

pooledBin(x=x1, m=m1, n=n1)

summary(pooledBin(x=x1, m=m1, n=n1), scale=1000)

# For another population, tested with the same design, one might find:
# 1 positive result among the pools pooling 5 elements,
# no positive result among the pools pooling 10 elements,
# 4 positive results among the pools pooling 50 elements,

x2<-c(0,1,0,4)
m2 <- c(1,5,10,50)
n2 <- c(5,5,5,5)

pooledBin(x=x2, m=m2, n=n2)

# Some other methods for the confidence bounds:

pooledBin(x=x2, m=m2, n=n2, ci.method="lrt")


###
# Reproducing some of the estimates from Table 1 in
# Hepworth & Biggerstaff (2017):

pooledBin(x=c(1,2), m=c(20,5), n=c(8,8), pt.method="firth", ci.method="lrt")

pooledBin(x=c(7,8), m=c(20,5), n=c(8,8), pt.method="firth", ci.method="lrt")
```

`pooledBinDiff` *Confidence intervals for the difference of proportions*

Description

The function calculates confidence intervals for the difference of two proportions based on pooled testing experiments containing various different pool sizes.

Usage

```
pooledBinDiff(x1, m1, x2, m2, n1 = rep(1, length(x1)),
 n2 = rep(1, length(x2)),
 pt.method = c("firth", "gart", "bc-mle", "mle", "mir"),
 ci.method = c("skew-score", "bc-skew-score", "score", "lrt", "wald", "mir"),
 scale = 1, alpha = 0.05, tol = .Machine$double.eps^0.5)
```

Arguments

x1	a vector, specifying the observed number of positive pools, among the number of pools tested (n1) in population 1
m1	a vector of corresponding pool sizes in population 1, must have the same length as x1
x2	a vector, specifying the observed number of positive pools, among the number of pools tested (n2) in population 2
m2	a vector of corresponding pool sizes in population 2, must have the same length as x2
n1	a vector of the corresponding number of pools with sizes m1
n2	a vector of the corresponding number of pools with sizes m2
pt.method	a character string, specifying the point estimator to compute, with the following options: "firth": recommended bias-corrected estimator (Hepworth & Biggerstaff, 2017), the default; "gart" bias-corrected MLE (Biggerstaff, 2008); "bc-mle": same as "gart", kept for backward compatibility ; "mle" MLE, and "mir" MIR.
ci.method	a character string, specifying the confidence interval to compute, with options: "skew-score" skewness-corrected, the default, "score" the score, "bc-skew-score" bias- and skewness-corrected "lrt" likelihood ratio test. See Biggerstaff (2008) for details. Further, the options "wald" wald and "mir" MIR are available but are not recommended.
scale	a single numeric, coefficient to scale the point estimates and intervals bounds in the print and summary method (print.poolbindiff, summary.poolbindiff)
alpha	a single numeric, specifying the type-I-error level
tol	accuracy required for iterations in internal functions

Details

Point estimation: bias preventative ("firth") and bias corrected ("gart") estimators are recommended, with details described in Hepworth G, Biggerstaff BJ (2017). Use of MLE ("mle"), and MIR ("mir") estimators is not recommended.

Confidence intervals: Note, that the methods "mir" and "wald" can not be recommended, because they return too narrow intervals in relevant situations, "mir" because it ignores the pooling, and "wald" because it relies on simple large sample methods. For computational details and simulation results of the remaining methods, see Biggerstaff (2008).

Value

A list with elements

d	the estimated difference of proportions
lcl	the lower confidence limit
ucl	the upper confidence limit
pt.method	the method used for point estimation

ci.method	the method used for interval estimation
alpha	the type-I-error level
scale	Scaling coefficient for the output
x1	the numbers of postive pools in population 1
m1	the size of the pools in population 1
n1	the numbers of pools with corresponding pool sizes m1 in population 1
x2	the numbers of postive pools in population 2
m2	the size of the pools in population 2
n2	the numbers of pools with corresponding pool sizes m1 in population 2

Author(s)

Brad Biggerstaff

References

Walter SD, Hildreth SW, Beaty BJ: Estimation of infection rates in population of organisms using pools of variable size. *Am J Epidemiol 1980, 112(1):124-128*

Hepworth G: Estimation of proportions by group testing. PhD Dissertation. Melbourne, Australia: The University of Melbourne; 1999.

Biggerstaff BJ (2008): Confidence interval for the difference of proportions estmimated from pooled samples. *Journal of Agricultural Biological and Environmental Statistics 2008, 13(4):478-496.*

Hepworth G, Biggerstaff BJ: Bias correction in estimating Proportions by pooled testing. *JABES 2017, to appear.*

Examples

```
# Consider an imaginary example, where pools of size
# 1, 5, 10 and 50 are tested, 5 pools of each size. The same
# design is applied to estimate the prevalence in
# two populations:
# In population 1, among the 5 pools with size 1 and 5,
# no positive pool is observed,
# while among the 5 pools of size 10 and 50, 1 and 2 positive
# pools are identified, respectively.

# In population 2, 1 positive result is observed
# among the 5 pools each pooling 5 elements,
# no positive result among the pools pooling 10 elements,
# 4 positive results among the pools pooling 50 elements,

x1 <- c(0,0,1,2)
m <- c(1,5,10,50)
n <- c(5,5,5,5)

x2<-c(0,1,0,4)
```

```
pooledBinDiff(x1=x1, m1=m, x2=x2, m2=m, n1=n, n2=n)

summary(pooledBinDiff(x1=x1, m1=m, x2=x2, m2= m, n1=n, n2=n), scale=1000)

# Compare recommended methods:

pooledBinDiff(x1=x1, m1=m, x2=x2, m2= m, n1=n, n2=n, pt.method="mle",
 ci.method="lrt")

pooledBinDiff(x1=x1, m1=m, x2=x2, m2= m, n1=n, n2=n, pt.method="mle",
 ci.method="score")

pooledBinDiff(x1=x1, m1=m, x2=x2, m2= m, n1=n, n2=n, pt.method="mle",
 ci.method="skew-score")
```

`predict.gt` *Predict Method for Group Testing Model Fits*

Description

Obtains predictions for individual observations and optionally estimates standard errors of those predictions from objects of class `"gt"` or `"gt.mp"` returned by `gtreg` and `gtreg.mp`, respectively.

Usage

```
## S3 method for class 'gt'
predict(object, newdata, type = c("link", "response"),
 se.fit = FALSE, conf.level = NULL, na.action = na.pass, ...)
```

Arguments

`object`	a fitted object of class `"gt"` or `"gt.mp"`.
`newdata`	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`type`	the type of prediction required. The option `"link"` is on the scale of the linear predictors; the alternative `"response"` is on the scale of the response variable. Thus for the binomial model the `"link"` predictions are of log-odds (probabilities on logit scale) and `type = "response"` gives the predicted probabilities.
`se.fit`	logical switch indicating if standard errors are required.
`conf.level`	confidence level of the interval of the predicted values.
`na.action`	function determining what should be done with missing values in `newdata`. The default is to predict `NA`.
`...`	currently not used

Details

If newdata is omitted the predictions are based on the data used for the fit. When newdata is present and contains missing values, how the missing values will be dealt with is determined by the na.action argument. In this case, if na.action = na.omit omitted cases will not appear, whereas if na.action = na.exclude they will appear (in predictions and standard errors), with value NA. See also napredict.

Value

If se = FALSE, a vector or matrix of predictions. If se = TRUE, a list with components

fit	Predictions
se.fit	Estimated standard errors
lower	Lower bound of the confidence interval if calculated
upper	Upper bound of the confidence interval if calculated

Author(s)

Boan Zhang

Examples

```
data(hivsurv)

fit1 <- gtreg(formula = groupres ~ AGE + EDUC., data = hivsurv,
  groupn = gnum, sens = 0.9, spec = 0.9, linkf = "logit", method = "V")
pred.data <- data.frame(AGE = c(15, 25, 30), EDUC. = c(1, 3, 2))
predict(object = fit1, newdata = pred.data, type = "link", se.fit = TRUE)
predict(object = fit1, newdata = pred.data, type = "response",
  se.fit = TRUE, conf.level = 0.9)
predict(object = fit1, type = "response", se.fit = TRUE, conf.level = 0.9)
```

---

print.bgt *Print Functions for Group Testing CIs and Tests for One Proportion*

Description

Print objects of class "bgtCI", "bgtTest", "bgtvs", "binCI", and "binTest"

Usage

```
## S3 method for class 'bgtCI'
print(x, ...)
## S3 method for class 'binCI'
print(x, ...)
## S3 method for class 'bgtTest'
print(x, ...)
## S3 method for class 'binTest'
print(x, ...)
## S3 method for class 'bgtvs'
print(x, ...)
```

Arguments

x	an object of the corresponding class
...	currently only digits is passed to signif for appropriate rounding

Value

A print out.

---

print.bgtDesign *Print Functions for nDesign and sDesign*

Description

Print function for results of functions nDesign, nDesign.

Usage

```
## S3 method for class 'nDesign'
print(x, ...)
## S3 method for class 'sDesign'
print(x, ...)
```

Arguments

x	An object of class "nDesign" or "sDesign"
...	currently only digits is passed to signif for appropriate rounding

print.binDesign *Print Function for binDesign*

Description

Print objects of class "binDesign"

Usage

```
## S3 method for class 'binDesign'
print(x, ...)
```

Arguments

x	an object of class "binDesign"
...	currently only digits is passed to signif for appropriate rounding

print.gt *Print methods for objects of classes "gt" and "gt.mp"*

Description

Print methods for objects of classes "gt" and "gt.mp"

Usage

```
## S3 method for class 'gt'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	objects of class "gt" and "gt.mp", respectively
digits	the digits for rounding
...	currently not used

Value

A print out

print.poolbindiff *Print methods for classes "poolbin" and "poolbindiff"*

Description

Print methods for objects of classes "poolbin" and "poolbindiff"

Usage

```
## S3 method for class 'poolbin'
print(x, scale = x$scale, ...)
## S3 method for class 'poolbindiff'
print(x, scale = x$scale, ...)
```

Arguments

x	An object of class "poolbin" or "poolbindiff" (pooledBin, pooledBinDiff)
scale	A coefficient to scale the point estimate and interval bounds
...	further arguments to be passed to print

Value

A print out.

print.summary.gt *Print Functions for summary.gt.mp and summary.gt*

Description

Print function for objects obtained by calling summary.gt.mp and summary.gt

Usage

```
## S3 method for class 'summary.gt.mp'
print(x, digits = max(3, getOption("digits") - 3),
 signif.stars = getOption("show.signif.stars"), ...)
## S3 method for class 'summary.gt'
print(x, digits = max(3, getOption("digits") - 3),
 signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

x	an object of class "summary.gt.mp" or "summary.gt"
digits	digits for rounding
signif.stars	logical, indicating whether significance stars should be shown
...	further arguments to be passed to printCoefmat

Value

A print out.

`residuals.gt` *Extract Model Residuals From a Fitted Group Testing Model*

Description

`residuals.gt` is a function which extracts model residuals from objects of class `"gt"` returned by `gtreg`.

Usage

```
## S3 method for class 'gt'
residuals(object, type = c("deviance", "pearson", "response"),...)
```

Arguments

`object`	an object of class `"gt"` from which the model residuals are to be extracted.
`type`	the type of residuals which should be returned. The alternatives are: `"deviance"` (default), `"pearson"`, and `"response"`.
`...`	currently not used

Value

Residuals of group responses extracted from the object `object`.

Author(s)

Boan Zhang

Examples

```
data(hivsurv)

fit1 <- gtreg(formula = groupres ~ AGE * EDUC., data = hivsurv, groupn = gnum,
              linkf = "probit")
residuals.gt(object = fit1, type = "pearson")
residuals.gt(object = fit1, type = "deviance")
```

sDesign *Iterate Group Size for a One-Parameter Group Testing Problem*

Description

Increasing of group size s for a fixed number of groups n in a binomial group testing design, until a pre-specified power is achieved. Control of bias during iteration. A hypothetical threshold proportion `p.hyp` and the absolute difference `delta` to be detected have to be specified.

Usage

```
sDesign(n, smax, delta, p.hyp, conf.level = 0.95,
 power = 0.8, alternative = "two.sided", method = "CP", biasrest = 0.05)
```

Arguments

`n`	integer, fixed sample size (number of assays)
`smax`	integer, maximal group size allowed in planning of the design
`delta`	absolute difference between the threshold and the true proportion which shall be detectable with specified power
`p.hyp`	threshold proportion to test against in the hypothesis, specify as a value between 0 and 1
`conf.level`	confidence level of the decision, default is 0.95
`power`	level of power to be achieved to be specified as a probability between 0 and 1
`alternative`	character string, defining the alternative hypothesis, either 'two.sided', 'less' or 'greater' where `'less'` calculates the probability that p.hyp is excluded by an upper confidence limit for a true proportion `p.hyp - delta`, `'greater'` calculates the probability that p.hyp is excluded by a lower confidence limit for a true proportion of `p.hyp + delta` . `'two.sided'` calculates min(power(`p.hyp - delta`, `p.hyp + delta`)) for a two.sided CI, thus can result in much lower power. Note that coverage probability and power are not necessarily symmetric for upper and lower bound of binomial CI.
`method`	character string specifying the CImethod to be used for evaluation, see argument `method` in `bgtCI`
`biasrest`	value between 0 and 1 specifying the absolute bias maximally allowed

Details

The power of a confidence interval here is defined as the probability that a confidence interval or limit excludes the threshold parameter (p.hyp) of the hypothesis. This function increases size of groups (number of units in a bulk sample) until a pre-specified power is reached. Since the power does not increase monotone with increasing s for binomial proportions but oscillates between local maxima and minima, the simple iteration given here will generally result in selecting those s, for which the given CI method shows a local minimum of coverage if the null hypothesis is true. Since the positive bias of the estimator in group testing increases with increasing group size, it is checked

whether bias is smaller than a pre-specified value (bias.rest). If bias violates this restriction for a given combination n, s, delta, s will not be further increased the actual power of the last acceptable group size s is returned.

Value

A list containing:

`sout`	the group size necessary to met the conditions
`powerout`	the exact power for the specified parameters and the group size
`biasout`	the bias for the specified parameters and the iterated group size

and a number of values specified in the function call or produced in the iteration.

References

Schaarschmidt F (2007) Experimental design for one-sided confidence intervals or hypothesis tests in binomial group testing. *Communications in Biometry and Crop Science 2 (1), 32-40.* http://agrobiol.sggw.waw.pl/cbcs/

Swallow WH (1985) Group testing for estimating infection rates and probabilities of disease transmission. *Phytopathology 75 (8), 882-889.*

See Also

`plot.sDesign` can be used to plot th iteration of this function

`bgtPower`: calculation of power and bias depending on n, s, delta, p.hyp, conf.level, method
`nDesign`: function to iteratively optimize sample size(number of groups)n for a given group size s
`estDesign`: function to choose group size s according to the minimal mse of the estimator, as given in Swallow (1985)

Examples

```
## Assume that objective is to show that a proportion
## is smaller than 0.005 (i.e. 0.5%) with a
## power of 0.80 (i.e. 80%) if the unknown proportion
## in the population is 0.003 (i.e. 0.3%), thus a
## delta = 0.002 shall be detected. A 95-per-cent
## Clopper-Pearson CI (corresponding to an exact test)
## shall be used. The maximal number of groups might
## be 30 where the assay sensitivity is not limited
## until groupsize s = 100.

sDesign(smax=100, n=30, delta=0.002, p.hyp=0.005,
 alternative="less", method="CP", power=0.8)

## One might accept to detect delta=0.004,
## i.e. accepting to reject H0: p>=0.005 with
## power 80 per cent when the true proportion is 0.001:

sDesign(smax=100, n=30, delta=0.004, p.hyp=0.005,
 alternative="less", method="CP", power=0.8)
```

```
sDesign(smax=100, n=30, delta=0.004, p.hyp=0.005,
 alternative="less", method="Score", power=0.8)
```

`sim.gt` *Simulation Function for Group Testing Data*

Description

Simulates data in group testing form ready to be fit by `gtreg`.

Usage

```
sim.gt(x = NULL, gshape = 20, gscale = 2, par,
     linkf = c("logit", "probit", "cloglog"),
     sample.size, group.size, sens = 1, spec = 1,
     sens.ind = NULL, spec.ind = NULL)
```

Arguments

`x`	a matrix of user-submitted covariates to simulate the data with, defaults to `NULL` in which case a gamma distribution is used to generate the covariates automatically
`gshape`	shape parameter of gamma distribution, must be non-negative, set to be 20 by default
`gscale`	scale parameter of gamma distribution, must be strictly positive, set to be 2 by default
`par`	the true coefficients in the linear predictor
`sample.size`	sample size of simulated data
`linkf`	a character string specifying one of the three link functions to be used: `"logit"` (default) or `"probit"` or `"cloglog"`
`group.size`	group size in pooling individual samples
`sens`	sensitivity of the group tests, set to be 1 by default.
`spec`	specificity of the group tests, set to be 1 by default.
`sens.ind`	sensitivity of the individual retests, set to be equal to `sens` if not specified otherwise.
`spec.ind`	specificity of the individual retests, set to be equal to `spec` if not specified otherwise.

Details

`sim.gt` generates group testing data in simple pooling form. The covariates are either specified by the `x` argument or they are generated from a gamma distribution with a given `gshape` and `gscale`. The individual probabilities are calculated from the covariates, the coefficients given in `par`, and the link function specified through `linkf`. The true binary individual responses are then simulated from the individual probabilities. The true group responses are found from the individual responses within the groups (i.e., if at least one response is positive, the group is positive; otherwise, the group response is negative). Finally, the observed group responses are simulated using the given `sens` and `spec`. Individual retests are simulated from `sens.ind` and `spec.ind` for samples in observed positive groups. Note that with a given `group.size`, the last group may have less individuals.

Value

`sim.gt` returns a data frame with the following columns:

`gres`	the group response
`x`	the covariate
`groupn`	the group number
`ind`	the actual individual response
`retest`	the results of individual retests

Author(s)

Boan Zhang

See Also

`gtreg`, `gtreg.mp`

Examples

```
set.seed(46)
gt.data <- sim.gt(par = c(-12, 0.2), sample.size = 700, group.size = 5)

x1 <- sort(runif(100, 0, 30))
x2 <- rgamma(100, shape=17, scale=1.5)
gt.data <- sim.gt(x = cbind(x1,x2), par=c(-14, 0.2, 0.3),
 group.size = 4, sens = 0.98, spec = 0.98)
```

`sim.halving` *Simulation Function for Group Testing Data for the Halving Protocol*

Description

Simulates data in group testing form ready to be fit by `gtreg.halving`.

Usage

```
sim.halving(x = NULL, gshape = 20, gscale = 2, par,
     linkf = c("logit", "probit", "cloglog"),
     sample.size, group.size, sens = 1, spec = 1,
     sens.ind = NULL, spec.ind = NULL)
```

Arguments

`x`	a matrix of user-submitted covariates to simulate the data with, defaults to `NULL` in which case a gamma distribution is used to generate the covariates automatically
`gshape`	shape parameter of gamma distribution, must be non-negative, set to be 20 by default
`gscale`	scale parameter of gamma distribution, must be strictly positive, set to be 2 by default
`par`	the true coefficients in the linear predictor
`sample.size`	sample size of simulated data
`linkf`	a character string specifying one of the three link functions to be used: `"logit"` (default) or `"probit"` or `"cloglog"`
`group.size`	group size in pooling individual samples
`sens`	sensitivity of the group tests, set to be 1 by default.
`spec`	specificity of the group tests, set to be 1 by default.
`sens.ind`	sensitivity of the individual retests, set to be equal to `sens` if not specified otherwise.
`spec.ind`	specificity of the individual retests, set to be equal to `spec` if not specified otherwise.

Details

`sim.halving` generates group testing data for the halving protocol. The covariates are either specified by the `x` argument or they are generated from a gamma distribution with a given `gshape` and `gscale`. The individual probabilities are calculated from the covariates, the coefficients given in `par`, and the link function specified through `linkf`. The true binary individual responses are then simulated from the individual probabilities. The group, subgroup, and individual retests are simulated using the given `sens` and `spec` under the halving protocol.

Value

`sim.halving` returns a data frame with the following columns:

`gres`	the group response
`x`	the covariate
`groupn`	the group number
`ind`	the actual individual response
`retest`	the results of individual retests
`subgroup`	the subgroup number

Author(s)

Boan Zhang

See Also

`gtreg.halving`

Examples

```
set.seed(46)
gt.data <- sim.halving(par = c(-6, .1), gshape = 17, gscale = 1.4,
               sample.size = 5000, group.size = 5,
               sens = 0.95, spec = 0.95)
```

---

sim.mp *Simulation Function for Group Testing Data with Matrix Pooling Design*

Description

Simulates data in group testing form ready to be fit by `gtreg.mp`.

Usage

```
sim.mp(x = NULL, gshape = 20, gscale = 2, par,
    linkf = c("logit", "probit", "cloglog"),
    n.row, n.col, sens = 1, spec = 1,
    sens.ind = NULL, spec.ind = NULL)
```

Arguments

x	a matrix of user-submitted covariates to simulate the data with, defaults to NULL in which case a gamma distribution is used to generate the covariates automatically.
gshape	shape parameter of gamma distribution, must be non-negative, set to be 20 by default
gscale	scale parameter of gamma distribution, must be strictly positive, set to be 2 by default
par	the true coefficients in the linear predictor
linkf	a character string specifying one of the three link functions to be used: "logit" (default) or "probit" or "cloglog"
n.row	a vector that specifies the number of rows in each matrix, a scalar if only one matrix is simulated
n.col	a vector that specifies the number of columns in each matrix, a scalar if only one matrix is simulated
sens	sensitivity of the group tests, set to be 1 by default.
spec	specificity of the group tests, set to be 1 by default.
sens.ind	sensitivity of the individual retests, set to be equal to sens if not specified otherwise.
spec.ind	specificity of the individual retests, set to be equal to spec if not specified otherwise.

Details

sim.mp generates group testing data in matrix pooling form. The covariates are either specified by the x argument or they are generated from a gamma distribution with a given gshape and gscale. The individual probabilities are calculated from the covariates, the coefficients given in par and the link function specified through linkf. The true binary individual responses are then simulated from the individual probabilities. The individuals are organized into (by column) one or more matrices specified by n.row and n.col, and the true group responses are found (i.e., if at least one response is positive, the group is positive; otherwise, the group response is negative). The observed row and column group responses are then simulated using the given sens and spec values. Individual retests are simulated from sens.ind and spec.ind for individuals that lie on the intersection of an observed positive row and an observed positive column. In the case where no column (row) tests positive in a matrix, all the individuals in any observed positive rows (columns) will be assigned a simulated retest result. If no column or row is observed positive, NULL is returned.

Value

sim.mp returns a list with the components dframe: the data frame that is actually to be fit, ind: the true individual responses presented in matrices and prob: the individual probabilities.

dframe is a data frame with columns

col.resp	the column group response
row.resp	the row group response

x	the covariate
arrayn	the array number
coln	the column group number
rown	the row group number
retest	the results of individual retests

Author(s)

Boan Zhang

See Also

gtreg.mp for the corresponding function to fit the model.

Examples

```
# 5*6 and 4*5 matrix
set.seed(9128)
sa1a<-sim.mp(par=c(-7,0.1), n.row=c(5,4), n.col=c(6,5),
 sens=0.95, spec=0.95)
sa1<-sa1a$dframe
```

---

summary.gt *Summary Method for Group Testing Model (Simple Pooling) Fits*

Description

Produce a summary list for objects of class "gt" returned by gtreg.

Usage

```
## S3 method for class 'gt'
summary(object, ...)
```

Arguments

object	a fitted object of class "gt".
...	currently not used.

Details

print.summary.gt is the print function that formats the coefficients, standard errors, etc. and additionally gives 'significance stars' if signif.stars is TRUE. The coefficients component of the result gives the estimated coefficients and their estimated standard errors, together with their ratio. This third column is labeled z ratio using Wald tests. A fourth column gives the two-tailed p-value corresponding to the z ratio based on a Wald test. (It is possible that there are no residual degrees of freedom from which to estimate it. In that case the estimate is NaN.)

Value

summary.gt returns an object of class "summary.gt", a list with components

call	the component from object.
link	the component from object.
deviance	the component from object.
aic	the component from object.
df.residual	the component from object.
null.deviance	the component from object.
df.null	the component from object.
deviance.resid	the deviance residuals: see residuals.gt
coefficients	the matrix of coefficients, standard errors, z-values and p-values. Aliased coefficients are omitted.
counts	the component from object.
method	the component from object.
cov.mat	the estimated covariance matrix of the estimated coefficients.

Author(s)

Boan Zhang

See Also

gtreg for creating an object of class "gt", and print.summary.gt for some options of changing the print out following summary.gt.

Examples

```
## --- Continuing the Example from '?gtreg':

data(hivsurv)

fit1 <- gtreg(formula = groupres ~ AGE + EDUC., data = hivsurv,
  groupn = gnum, sens = 0.9, spec = 0.9, method = "Xie")

summary(fit1)
```

summary.gt.mp *Summary Method for Group Testing Model (Matrix Pooling) Fits*

Description

Produce a summary list for objects of class "gt.mp" returned by gtreg.mp.

Usage

```
## S3 method for class 'gt.mp'
summary(object, ...)
```

Arguments

object	a fitted object of class "gt.mp".
...	currently not used.

Details

print.summary.gt.mp is the print function that formats the coefficients, standard errors, etc. and additionally gives 'significance stars' if signif.stars is TRUE. The coefficients component of the result gives the estimated coefficients and their estimated standard errors, together with their ratio. This third column is labeled z ratio using Wald tests. A fourth column gives the two-tailed p-value corresponding to the z ratio based on a Wald test.

Value

summary.gt.mp returns an object of class "summary.gt.mp", a list with components

call	the component from object.
link	the component from object.
coefficients	the matrix of coefficients, standard errors, z-values and p-values.
counts	the component from object.
Gibbs.sample.size	the component from object.
cov.mat	the estimated covariance matrix of the estimated coefficients.

Author(s)

Boan Zhang

See Also

gtreg.mp for creating an object of class "gtreg.mp", print.summary.gt.mp gives some hints how to change the print-out of summary.gt.mp

Examples

```
## --- Continuing the Example from  '?sim.mp' and '?gtreg.mp':
# 5*6 and 4*5 matrix

set.seed(9128)
sa1a<-sim.mp(par=c(-7,0.1), n.row=c(5,4), n.col=c(6,5),
  sens=0.95, spec=0.95)
sa1<-sa1a$dframe

## Not run:
fit1mp <- gtreg.mp(formula = cbind(col.resp, row.resp) ~ x, data = sa1,
  coln = coln, rown = rown, arrayn = arrayn,
  sens = 0.95, spec = 0.95, linkf = "logit", n.gibbs = 1000, tol = 0.005)

summary(fit1mp)

## End(Not run)
```

`summary.poolbindiff` *Summary methods for "poolbin" and "poolbindiff"*

Description

Summary Method for One-Sample and Two-sample confidence intervals (various pool sizes)

Usage

```
## S3 method for class 'poolbin'
summary(object, scale = object$scale, ...)
## S3 method for class 'poolbindiff'
summary(object, scale = object$scale, ...)
```

Arguments

`object`	An object of class "poolbin" or "poolbindiff" (`pooledBin`, `pooledBinDiff`)
`scale`	A coefficient to scale the point estimate and interval bounds
`...`	further arguments to be passed to `print`

Value

A print out

`thresh.val.dorf` *Find the optimal threshold value for Thresholded Optimal Dorfman testing*

Description

Find the optimal threshold value for Thresholded Optimal Dorfman (TOD) testing.

Usage

```
thresh.val.dorf(p, psz, se = 1, sp = 1)
```

Arguments

`p`	a vector of each individual's probability of infection.
`psz`	the initial pool size.
`se`	the sensitivity of the diagnostic test.
`sp`	the specificity of the diagnostic test.

Details

This function finds the optimal threshold value for TOD testing for situations where the threshold value is not specified. See McMahan et al. (2012) for additional details on the implementation of TOD testing.

Value

The optimal threshold value for TOD testing.

Author(s)

This function was originally written by Christopher S. McMahan for McMahan et al. (2012). The function was obtained from http://chrisbilder.com/grouptesting/.

References

McMahan, C., Tebbs, J., Bilder, C. (2012). "Informative Dorfman Screening." *Biometrics*, **68**(1), 287–296. ISSN 0006341X, doi:10.1111/j.15410420.2011.01644.x, https://pubmed.ncbi.nlm.nih.gov/21762119/.

See Also

http://chrisbilder.com/grouptesting/

Other Informative Dorfman functions: `accuracy.dorf()`, `characteristics.pool()`, `inf.dorf.measures()`, `opt.info.dorf()`, `opt.pool.size()`, `pool.specific.dorf()`

Examples

```
# This example takes approximately 4 seconds to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
## Not run:
set.seed(3154)
p.vec <- p.vec.func(p=0.10, alpha=0.5, grp.sz=1000)
thresh.val.dorf(p=p.vec, psz=10, se=0.95, sp=0.95)
## End(Not run)

# This example takes less than 1 second to run.
# Estimated running time was calculated using a
#   computer with 16 GB of RAM and one core of an
#   Intel i7-6500U processor.
p.vec <- p.vec.func(p=0.15, alpha=2, grp.sz=100)
thresh.val.dorf(p=p.vec, psz=10, se=0.95, sp=0.95)
```

Index