Package 'groupTesting'

Title Simulating and Modeling Group (Pooled) Testing Data

Version 1.3.0

Date 2024-08-01

Maintainer Md S. Warasi <msarker@radford.edu>

Description Provides an expectation-maximization (EM) algorithm using the approach introduced in Warasi (2023) <doi:10.1080/03610918.2021.2009867>. The EM algorithm can be used to estimate the prevalence (overall proportion) of a disease and to estimate a binary regression model from among the class of generalized linear models based on group testing data. The estimation framework we consider offers a flexible and general approach; i.e., its application is not limited to any specific group testing protocol. Consequently, the EM algorithm can model data arising from simple pooling as well as advanced pooling such as hierarchical testing, array testing, and quality control pooling. Also, provided are functions that can be used to conduct the Wald tests described in Buse (1982) <doi:10.1080/00031305.1982.10482817> and to simulate the group testing data described in Kim et al. (2007) <doi:10.1111/j.1541-0420.2007.00817.x>. We offer a function to compute relative efficiency measures, which can be used to optimize the maximum likelihood estimator of disease prevalence.

License GPL-3

Depends R(>= 3.5.0)

Imports binGroup2, pracma, stats, parallel

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.3.1

Author Md S. Warasi [aut, cre]

Repository CRAN

Date/Publication 2024-08-17 15:20:02 UTC

Contents

---

array.gt.simulation   *Simulating Array-Based Group Testing Data*

Description

This function simulates two-dimensional array-based group testing data.

Usage

```
array.gt.simulation(
  N,
  p = 0.1,
  protocol = c("A2", "A2M"),
  n,
  Se,
  Sp,
  assayID,
  Yt = NULL
)
```

Arguments

N	The number of individuals to be tested.
p	A vector of length N consisting of individual disease probabilities.
protocol	Either "A2" or "A2M", where "A2" ("A2M") refers to the two-dimensional array without (with) testing the members of an array as a single pooled sample.
n	The row (or column) size of the arrays.
Se	A vector of assay sensitivities.
Sp	A vector of assay specificities.
assayID	A vector of assay identification numbers.
Yt	A vector of individual true disease statuses.

Details

We consider the array testing protocol outlined in Kim et al. (2007). Under this protocol, N individuals are assigned to m non-overlapping n-by-n matrices such that $N = mn^2$. From each matrix, n pools are formed using the row specimens and another n pools are formed using the column specimens. In stage 1, the $2n$ pools are tested. In stage 2, individual testing is used for case identification according to the strategy described in Kim et al. (2007). This is a 2-stage protocol called *Square Array without Master Pool Testing* and denoted by $A2(n : 1)$ in Kim et al. (2007). A variant (3-stage protocol) is also presented in Kim et al. (2007) which employs testing the n^2 array members together as an initial pooled unit before implementing the 2-stage array. If the initial pooled test is negative, the procedure stops (i.e., the 2-stage array is not needed). However, if the pooled test is positive, the 2-stage protocol is used as before. This 3-stage approach is called *Square Array with Master Pool Testing* and is denoted by $A2(n^2 : n : 1)$. See Kim et al. (2007) for more details.

N should be divisible by the array size n^2. When not divisible, the remainder individuals are tested one by one (i.e., individual testing).

p is a vector of individual disease probabilities. When all individuals have the same probability of disease, say, 0.10, p can be specified as rep(0.10, N) or p=0.10.

For "A2" and "A2M", the pool sizes used are c(n, 1) and c(n^2, n, 1), respectively.

For "A2", Se is c(Se1, Se2), where Se1 is the sensitivity of the assay used for both row and column pools, and Se2 is the sensitivity of the assay used for individual testing. For "A2M", Se is c(Se1, Se2, Se3), where Se1 is for the initial array pool, Se2 is for the row and column pools, and Se3 is for individual testing. Sp is specified in the same manner.

For "A2", assayID is c(1, 1) when the same assay is used for row/column pool testing as well as for individual testing, and assayID is c(1, 2) when assay 1 is used for row/column pool testing and assay 2 is used for individual testing. In the same manner, assayID is specified for "A2M" as c(1, 1, 1), c(1, 2, 3), and in many other ways.

When available, the individual true disease statuses (1 for positive and 0 for negative) can be used in simulating the group testing data through argument Yt. When an input is entered for Yt, argument p will be ignored.

Value

A list with components:

gtData	The simulated group testing data.
testsExp	The number of tests expended in the simulation.

References

Kim HY, Hudgens M, Dreyfuss J, Westreich D, and Pilcher C (2007). Comparison of Group Testing Algorithms for Case Identification in the Presence of Testing Error. *Biometrics*, 63(4), 1152–1163.

See Also

[hier.gt.simulation](#) for simulation of the hierarchical group testing data.

Examples

```
library(groupTesting)

## Example 1: Square Array without Master Pool Testing (i.e., 2-Stage Array)
N <- 48              # Sample size
protocol <- "A2"     # 2-stage array
n <- 4               # Row/column size
Se <- c(0.95, 0.95)  # Sensitivities in stages 1-2
Sp <- c(0.98, 0.98)  # Specificities in stages 1-2
assayID <- c(1, 1)   # The same assay in both stages

# (a) Homogeneous population
pHom <- 0.10         # Overall prevalence
array.gt.simulation(N=N,p=pHom,protocol=protocol,n=n,Se=Se,Sp=Sp,assayID=assayID)

# Alternatively, the individual true statuses can be used as:
yt <- rbinom( N, size=1, prob=0.1 )
array.gt.simulation(N=N,protocol=protocol,n=n,Se=Se,Sp=Sp,assayID=assayID,Yt=yt)

# (b) Heterogeneous population (regression)
param <- c(-3,2,1)
x1 <- rnorm(N, mean=0, sd=.75)
x2 <- rbinom(N, size=1, prob=0.5)
X <- cbind(1, x1, x2)
pReg <- exp(X%*%param)/(1+exp(X%*%param)) # Logit
array.gt.simulation(N=N,p=pReg,protocol=protocol,n=n,Se=Se,Sp=Sp,assayID=assayID)

# The above examples with different assays
Se <- c(0.95, 0.98)
Sp <- c(0.97, 0.99)
assayID <- c(1, 2)
array.gt.simulation(N,pHom,protocol,n,Se,Sp,assayID)
array.gt.simulation(N,pReg,protocol,n,Se,Sp,assayID)

## Example 2: Square Array with Master Pool Testing (i.e., 3-Stage Array)
N <- 48
protocol <- "A2M"
n <- 4
Se <- c(0.95, 0.95, 0.95)
Sp <- c(0.98, 0.98, 0.98)
assayID <- c(1, 1, 1)     # The same assay in 3 stages

# (a) Homogeneous population
pHom <- 0.10
array.gt.simulation(N,pHom,protocol,n,Se,Sp,assayID)

# (b) Heterogeneous population (regression)
param <- c(-3,2,1)
x1 <- rnorm(N, mean=0, sd=.75)
x2 <- rbinom(N, size=1, prob=0.5)
X <- cbind(1, x1, x2)
pReg <- exp(X%*%param)/(1+exp(X%*%param)) # Logit
```

```
array.gt.simulation(N,pReg,protocol,n,Se,Sp,assayID)

# The above examples with different assays:
Se <- c(0.95, 0.98, 0.98)
Sp <- c(0.97, 0.98, 0.92)
assayID <- 1:3
array.gt.simulation(N,pHom,protocol,n,Se,Sp,assayID)
array.gt.simulation(N,pReg,protocol,n,Se,Sp,assayID)
```

---

glm.gt   *EM Algorithm for Fitting Regression Models to Group Testing Data*

Description

This function implements an expectation-maximization (EM) algorithm to fit regression models to group testing data, where pooled responses are related to individual covariates through a link function in the generalized linear model (GLM) family. The EM algorithm, which is outlined in Warasi (2023), finds the maximum likelihood estimate (MLE) for the vector of regression coefficients, **beta**. The EM algorithm can model pooling data observed from **any** group testing protocol used in practice, including hierarchical and array testing (Kim et al., 2007).

Usage

```
glm.gt(
  beta0,
  gtData,
  X,
  g,
  dg = NULL,
  d2g = NULL,
  grdMethod = c("central", "forward", "backward"),
  covariance = FALSE,
  nburn = 2000,
  ngit = 5000,
  maxit = 200,
  tol = 0.001,
  tracing = TRUE,
  conf.level = 0.95,
  ...
)
```

Arguments

beta0	An initial value for the regression coefficients.
gtData	A matrix or data.frame consisting of the pooled test outcomes and other information from a group testing application. Needs to be specified as shown in the example below.

X	The design matrix.	
g	An inverse link function in the GLM family.	
dg	The first derivate of g. When NULL, a finite-difference approximation will be used.	
d2g	The second derivate of g. When NULL, a finite-difference approximation will be used.	
grdMethod	The finite-difference approximation method to be used for dg and d2g. See 'Details'.	
covariance	When TRUE, the covariance matrix is calculated at the MLE.	
nburn	The number of initial Gibbs iterates to be discarded.	
ngit	The number of Gibbs iterates to be used in the E-step after discarding nburn iterates as a burn-in period.	
maxit	The maximum number of EM steps (iterations) allowed in the EM algorithm.	
tol	Convergence tolerance used in the EM algorithm.	
tracing	When TRUE, progress in the EM algorithm is displayed.	
conf.level	Confidence level to be used for the Wald confidence interval.	
...	Further arguments to be passed to [optim](). See 'Details'.	

Details

gtData must be specified as follows. Columns 1-5 consist of the pooled test outcomes (0 for negative and 1 for positive), pool sizes, pool-specific sensitivities, pool-specific specificities, and assay identification (ID) numbers, respectively. From column 6 onward, the pool member ID numbers need to be specified. Note that the ID numbers must start with 1 and increase consecutively up to N, the total number of individuals tested. **For smaller pools, incomplete ID numbers must be filled out by -9 or any non-positive numbers** as shown in the example below. The design matrix X consists of invidual covariate information, such as age, sex, and symptoms, of the pool members located in column 6 onward.

	Z	psz	Se	Sp	Assay	Mem1	Mem2	Mem3	Mem4	Mem5	Mem6
Pool:1	1	6	0.90	0.92	1	1	2	3	4	5	6
Pool:2	0	6	0.90	0.92	1	7	8	9	10	11	12
Pool:3	1	2	0.95	0.96	2	1	2	-9	-9	-9	-9
Pool:4	0	2	0.95	0.96	2	3	4	-9	-9	-9	-9
Pool:5	1	2	0.95	0.96	2	5	6	-9	-9	-9	-9
Pool:6	0	1	0.92	0.90	3	1	-9	-9	-9	-9	-9
Pool:7	1	1	0.92	0.90	3	2	-9	-9	-9	-9	-9
Pool:8	0	1	0.92	0.90	3	5	-9	-9	-9	-9	-9
Pool:9	0	1	0.92	0.90	3	6	-9	-9	-9	-9	-9

This is an example of gtData, where 12 individuals are assigned to 2 non-overlapping initial pools and then tested based on the 3-stage hierarchical protocol. The test outcomes, Z, from 9 pools are in column 1. In three stages, different pool sizes (6, 2, and 1), sensitivities, specificities, and assays are used. The ID numbers of the pool members are shown in columns 6-11. The row names and

column names are not required. Note that the EM algorithm can accommodate any group testing data including those described in Kim et al. (2007). For individual testing data, the pool size in column 2 is 1 for all pools.

X is an $N\mathrm{x}k$ design matrix, where each column represents a vector of individual covariate values. For an intercept model, the first column values must be 1. The column (covariate) names of X, such as 'age' and 'sex', will be displayed in the estimation summary. When column names are missing (NULL), the names that will be displayed by default are 'Intercept', 'x1', 'x2', and so on.

The EM algorithm implements a Gibbs sampler to approximate the expectation in the E-step. Under each EM iteration, ngit Gibbs samples are retained for these purposes after discarding the initial nburn samples.

g relates the pooled responses Z (column 1 in gtData) to X. dg and d2g can be specified analogously. These characteristics can be obtained from glmLink for the common links: logit, probit, and complementary log-log.

grdMethod is used only when dg and d2g are NULL, where a finite-difference approximation is implemented by the function fderiv from the package 'pracma'.

The optimization routine optim is used to complete the M-step with the default method 'Nelder-Mead'. The argument ... allows the user to change the default method as well as other arguments in optim.

The covariance matrix is calculated by an appeal to the missing data principle and the method outlined in Louis (1982).

Value

A list with components:

param	The MLE of the regression coefficients.
covariance	Estimated covariance matrix for the regression coefficients.
iterUsed	The number of EM iterations needed for convergence.
convergence	0 if the EM algorithm converges successfully and 1 if the iteration limit maxit has been reached.
summary	Estimation summary with Wald confidence interval.

References

Kim HY, Hudgens M, Dreyfuss J, Westreich D, and Pilcher C. (2007). Comparison of Group Testing Algorithms for Case Identification in the Presence of Testing Error. *Biometrics*, 63:1152-1163.

Louis T. (1982). Finding the Observed Information Matrix when Using the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 44:226-233.

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.

Warasi M. (2023). groupTesting: An R Package for Group Testing Estimation. *Communications in Statistics-Simulation and Computation*, 52:6210-6224.

See Also

hier.gt.simulation and array.gt.simulation for group testing data simulation, and prop.gt for estimation of a disease prevalence from group testing data.

Examples

```
library(groupTesting)

## To illustrate 'glm.gt', we use data simulated
## by the functions 'hier.gt.simulation' and 'array.gt.simulation'.

## Note: The simulated data-structures are consistent
## with the data-structure required for 'gtData'.

## Example 1: MLE from 3-stage hierarchical group testing data.
## The data used is simulated by 'hier.gt.simulation'.

N <- 200              # Sample size
S <- 3                # 3-stage hierarchical testing
psz <- c(6,2,1)       # Pool sizes used in stages 1-3
Se <- c(.95,.95,.98)  # Sensitivities in stages 1-3
Sp <- c(.95,.98,.96)  # Specificities in stages 1-3
assayID <- c(1,2,3)   # Assays used in stages 1-3
param.t <- c(-3,2,1)  # The TRUE parameter to be estimated

# Simulating covariates:
set.seed(123)
x1 <- rnorm(N, mean=0, sd=0.75)
x2 <- rbinom(N, size=1, prob=0.5)
X <- cbind(1, x1, x2)
colnames( X ) <- c("Intercept", "Predictor 1", "Predictor 2")
# Note: Because the 1st column of X is 1, intercept model will be fit.

# Specifying logit inverse link:
g <- function(t){exp(t)/(1+exp(t))}
pReg <- g(X%*%param.t)

# Simulating test responses:
gtOut <- hier.gt.simulation(N,pReg,S,psz,Se,Sp,assayID)$gtData

# Fitting the model (with intercept):
param0 <- param.t + 0.2   # Initial value
res <- glm.gt(beta0=param0,gtData=gtOut,X=X,
              g=g,dg=NULL,d2g=NULL,
              grdMethod="central",covariance=TRUE,
              nburn=2000,ngit=5000,maxit=200,
              tol=1e-03,tracing=TRUE,conf.level=0.95)

# Note: Because dg and d2g are NULL (i.e., the exact derivatives
#       are not given), numerical derivatives are used.

# Estimation results:
# > res

# $param
# [1] -2.840802  1.992916  0.677176
```

```
# $covariance
#              [,1]        [,2]        [,3]
# [1,]  0.2134439 -0.10147555 -0.16693776
# [2,] -0.1014756  0.16855122  0.02997113
# [3,] -0.1669378  0.02997113  0.26324589

# $iterUsed
# [1] 10

# $convergence
# [1] 0

# $summary
#             Estimate Std.Err 95%lower 95%upper
# Intercept     -2.841   0.462   -3.746   -1.935
# Predictor 1    1.993   0.411    1.188    2.798
# Predictor 2    0.677   0.513   -0.328    1.683

## Example 2: MLE from two-dimensional array testing data.
## The data used is simulated by 'array.gt.simulation'.

N <- 200            # Sample size
protocol <- "A2"     # 2-stage array without testing initial master pool
n <- 5               # Row/column size
Se <- c(0.95, 0.95)  # Sensitivities
Sp <- c(0.98, 0.98)  # Specificities
assayID <- c(1, 1)   # The same assay in both stages
param <- c(-4,1,1)   # The TRUE parameter to be estimated

# Simulating data:
set.seed(123)
x1 <- runif(N)
x2 <- rnorm(N, mean=0, sd=0.5)
x3 <- rbinom(N, size=1, prob=0.5)
X <- cbind(x1, x2, x3)
# Note: Because the 1st column of X is not 1,
#       the model without intercept will be fit.

# Finding g, dg, and d2g from the function 'glmLink':
res0 <- glmLink(fn.name="logit")
g <- res0$g             # Logit inverse link g()
dg <- res0$dg           # The exact first derivate of g
d2g <- res0$d2g         # The exact second derivate of g
pReg <- g(X%*%param)    # Individual probabilities
gtOut <- array.gt.simulation(N,pReg,protocol,n,Se,Sp,assayID)$gtData

# Fitting the model (without intercept):
param0 <- param + 0.2
res <- glm.gt(beta0=param0,gtData=gtOut,X=X,g=g,
              dg=dg,d2g=d2g,covariance=TRUE,
              nburn=2000,ngit=5000,maxit=200,
              tol=1e-03,tracing=TRUE,conf.level=0.95)
print(res)
```

```
## Example 3: MLE from non-overlapping initial pooled responses.
## The data used is simulated by 'hier.gt.simulation'.

## Note: With initial pooled responses, our MLE is equivalent
## to the MLE in Vansteelandt et al. (2000).

N <- 1000              # Sample size
psz <- 5               # Pool size
S <- 1                 # 1-stage testing
Se <- 0.95             # Sensitivity
Sp <- 0.99             # Specificity
assayID <- 1           # Assay used for all pools
param <- c(-3,2,1)     # The TRUE parameter to be estimated

# Simulating data:
set.seed(123)
x1 <- rnorm(N, mean=0, sd=0.75)
x2 <- rbinom(N, size=1, prob=0.5)
X <- cbind(1, x1, x2)

# Finding g, dg, and d2g by the function 'glmLink':
res0 <- glmLink(fn.name="probit")  # Probit link
g <- res0$g
dg <- res0$dg
d2g <- res0$d2g
pReg <- g(X%*%param)
gtOut <- hier.gt.simulation(N,pReg,S,psz,Se,Sp,assayID)$gtData

# Fitting the model:
param0 <- param + 0.2
res <- glm.gt(beta0=param0,gtData=gtOut,X=X,g=g,
                  dg=dg,d2g=d2g,covariance=TRUE,
                  nburn=2000,ngit=5000,maxit=200,
                  tol=1e-03,tracing=TRUE,conf.level=0.95)
print(res)

## Example 4: MLE from individual (one-by-one) testing data.
## The data used is simulated by 'hier.gt.simulation'.

N <- 1000               # Sample size
psz <- 1                # Pool size 1 (i.e., individual testing)
S <- 1                  # 1-stage testing
Se <- 0.95              # Sensitivity
Sp <- 0.99              # Specificity
assayID <- 1            # Assay used for all pools
param <- c(-3,2,1)      # The TRUE parameter to be estimated

# Simulating data:
set.seed(123)
x1 <- rnorm(N, mean=0, sd=0.75)
x2 <- rbinom(N, size=1, prob=0.5)
```

```
X <- cbind(1, x1, x2)
g <- function(t){exp(t)/(1+exp(t))}  # Inverse logit
pReg <- g(X%*%param)
gtOut <- hier.gt.simulation(N,pReg,S,psz,Se,Sp,assayID)$gtData

# Fitting the model:
param0 <- param + 0.2
res <- glm.gt(beta0=param0,gtData=gtOut,
              X=X,g=g,dg=NULL,d2g=NULL,
              grdMethod="central",covariance=TRUE,
              nburn=2000,ngit=5000,maxit=200,
              tol=1e-03,tracing=TRUE,conf.level=0.95)
print(res)

## Example 5: Using pooled testing data.

# Pooled test outcomes:
Z <- c(1, 0, 1, 0, 1, 0, 1, 0, 0)

# Design matrix, X:
x1 <- c(0.8,1.2,0.4,1.5,1.8,1.8,0.1,1.6,0.2,0.2,1.8,0.2)
x2 <- c(31,56,45,64,26,47,22,60,35,41,32,41)
X <- cbind(x1, x2)

# Pool sizes used:
psz <- c(6, 6, 2, 2, 2, 1, 1, 1, 1)

# Pool-specific Se & Sp:
Se <- c(.90, .90, .95, .95, .95, .92, .92, .92, .92)
Sp <- c(.92, .92, .96, .96, .96, .90, .90, .90, .90)

# Assays used:
Assay <- c(1, 1, 2, 2, 2, 3, 3, 3, 3)

# Pool members:
Memb <- rbind(
   c(1, 2,  3,  4,  5,  6),
   c(7, 8,  9, 10, 11, 12),
   c(1, 2, -9, -9, -9, -9),
   c(3, 4, -9, -9, -9, -9),
   c(5, 6, -9, -9, -9, -9),
   c(1,-9, -9, -9, -9, -9),
   c(2,-9, -9, -9, -9, -9),
   c(5,-9, -9, -9, -9, -9),
   c(6,-9, -9, -9, -9, -9)
)
# The data-structure suited for 'gtData':
gtOut <- cbind(Z, psz, Se, Sp, Assay, Memb)

# Fitting the model with logit link:
g <- function(t){exp(t)/(1+exp(t))}
param0 <- c(0, 0)
res <- glm.gt(beta0=param0,gtData=gtOut,X=X,
```

```
                    g=g,dg=NULL,d2g=NULL,
                    grdMethod="central",covariance=TRUE,
                    nburn=2000,ngit=5000,maxit=200,
                    tol=1e-03,tracing=TRUE,conf.level=0.95)
     print(res)
```

---

glmLink   *Link Functions in the Class of Generalized Linear Models*

Description

This function provides characteristics of common link functions (logit, probit, and comlementary log-log). Specifically, based on the link name, the function with its inverse, first derivative, and second derivative is provided.

Usage

```
glmLink(fn.name = c("logit", "probit", "cloglog"))
```

Arguments

fn.name   One of the three: "logit", "probit", and "cloglog".

Value

A list with components:

g   The link function corresponding to "logit", "probit", or "cloglog".

dg   The first derivative of g.

d2g   The second derivative of g.

gInv   The inverse of g.

Examples

```
library(groupTesting)

## Try:
glmLink("logit")
```

---

hier.gt.simulation *Simulating Hierarchical Group Testing Data*

Description

This function simulates hierarchical group testing data with any number of hierarchical stages.

Usage

```
hier.gt.simulation(N, p = 0.1, S, psz, Se, Sp, assayID, Yt = NULL)
```

Arguments

N	The number of individuals to be tested.
p	A vector of length N consisting of individual disease probabilities.
S	The number of stages used in testing, where S >= 1.
psz	A vector of pool sizes in stages 1-S.
Se	A vector of assay sensitivities in stages 1-S.
Sp	A vector of assay specificities in stages 1-S.
assayID	A vector of the identification numbers of the assays used in stages 1-S.
Yt	A vector of individual true disease statuses.

Details

We consider the S-stage hierarchical testing protocol outlined in Kim et al. (2007). Under this protocol, N individual specimens are first assigned to m non-overlapping pools, where each initial pool size is c; i.e., $N = mc$. The initial pools are tested in stage 1. If a pooled test is negative, all members in the pool are diagnosed as negative. However, if a pooled test is positive, the pool members are split into non-overlapping subpools to be tested in the next stage. This procedure is continued. Note that individual testing is used in the final stage, S, for case identification.

S is a positive integer, S >= 1. When S=1, only the non-overlapping initial pools are tested in stage 1.

If N is not divisible by the initial pool size c, we implement the following policy to test the remainder individuals: (1) when S=1, simply test the remainder pool once as a pooled sample; (2) when S>1, test the remainder pool based on 2-stage hierarchical testing.

p is a vector of individual disease probabilities. When all individuals have the same probability of disease, say, 0.10, p can be specified as p=rep(0.10, N) or p=0.10.

psz is a vector of length S, where the first element is the stage-1 pool size, the second element is the stage-2 pool size, and so on. Pool size at any stage must be divisible by the pool size used at the next stage. For example, psz can be specified as c(12,3,1) but not as c(12,5,1).

When psz is a vector of length 1, test responses are simulated only from the initial pools.

Se is a vector of length S, where the first element is the sensitivity of the assay used in stage 1, the second element is sensitivity of the assay in stage 2, and so on.

Sp is a vector of length S, where the first element is the specificity of the assay used in stage 1, the second element is specificity of the assay in stage 2, and so on.

assayID is a vector of length S, where the first element is the ID of the assay in stage 1, the second element is the ID of the assay in stage 2, and so on.

When available, the individual true disease statuses (1 for positive and 0 for negative) can be used in simulating the group testing data through argument Yt. When an input is entered for Yt, argument p will be ignored.

Value

A list with components:

gtData   The simulated group testing data.

testsExp   The number of tests expended.

References

Kim HY, Hudgens M, Dreyfuss J, Westreich D, and Pilcher C (2007). Comparison of Group Testing Algorithms for Case Identification in the Presence of Testing Error. *Biometrics*, 63(4), 1152–1163.

See Also

[array.gt.simulation](#) for simulation of the array-based group testing data.

Examples

```
library(groupTesting)

## Example 1: Two-stage hierarchical (Dorfman) testing
N <- 50              # Sample size
psz <- c(5, 1)       # Pool sizes used in stages 1 and 2
S <- 2               # The number of stages
Se <- c(0.95, 0.95)  # Sensitivities in stages 1-2
Sp <- c(0.98, 0.98)  # Specificities in stages 1-2
assayID <- c(1, 1)   # The same assay in both stages

# (a) Homogeneous population
pHom <- 0.10         # Overall prevalence
hier.gt.simulation(N=N,p=pHom,S=S,psz=psz,Se=Se,Sp=Sp,assayID=assayID)

# Alternatively, the individual true statuses can be used as:
yt <- rbinom( N, size=1, prob=0.1 )
hier.gt.simulation(N=N,S=S,psz=psz,Se=Se,Sp=Sp,assayID=assayID,Yt=yt)

# (b) Heterogeneous population (regression)
param <- c(-3,2,1)
x1 <- rnorm(N, mean=0, sd=.75)
x2 <- rbinom(N, size=1, prob=0.5)
X <- cbind(1, x1, x2)
pReg <- exp(X%*%param)/(1+exp(X%*%param)) # Logit
hier.gt.simulation(N=N,p=pReg,S=S,psz=psz,Se=Se,Sp=Sp,assayID=assayID)
```

```
## Example 2: Initial (1-stage) pooled testing data
N <- 50
S <- 1
Se <- 0.95
Sp <- 0.98
assayID <- 1

# (a) Homogeneous population
pHom <- 0.10   # Overall prevalence

# a(i) Pooled testing
psz <- 5       # pool size
hier.gt.simulation(N,pHom,S,psz,Se,Sp,assayID)

# a(ii) Inidividual testing
psz <- 1       # pool size
hier.gt.simulation(N,pHom,S,psz,Se,Sp,assayID)

# (b) Heterogeneous population (regression)
param <- c(-3,2,1)
x1 <- rnorm(N, mean=0, sd=.75)
x2 <- rbinom(N, size=1, prob=0.5)
X <- cbind(1, x1, x2)
pReg <- exp(X%*%param)/(1+exp(X%*%param))  # Logit

# b(i) Pooled testing
psz <- 5
hier.gt.simulation(N,pReg,S,psz,Se,Sp,assayID)

# b(ii) Individual testing
psz <- 1
hier.gt.simulation(N,pReg,S,psz,Se,Sp,assayID)

## Example 3: Data with other configurations
N <- 48
p <- 0.10
Se <- c(.90, .95, .92, .90, .99)
Sp <- c(.96, .96, .90, .92, .95)
Assay <- 1:5

# Initial pooled testing, using the first element of Se, Sp & Assay
pszH1 <- 4
hier.gt.simulation(N=N,p=p,S=1,psz=pszH1,Se=Se,Sp=Sp,assayID=Assay)

pszH2 <- c(4,1)        # Two-stage, using first 2 elements of Se, Sp & Assay
hier.gt.simulation(N=N,p=p,S=2,psz=pszH2,Se=Se,Sp=Sp,assayID=Assay)

pszH4 <- c(16,8,2,1)  # Four-stage, using first 4 elements of Se, Sp & Assay
hier.gt.simulation(N=N,p=p,S=4,psz=pszH4,Se=Se,Sp=Sp,assayID=Assay)

pszH3 <- c(12,2,1)     # Three-stage, using first 3 elements of Se, Sp & Assay
Assay3 <- c(2,1,3)     # Array ID numbers do not need to be in order
```

```
hier.gt.simulation(N=N,p=p,S=3,psz=pszH3,Se=Se,Sp=Sp,assayID=Assay3)

# Works with a remainder pool of 2 individuals
N <- 50
psz <- c(12,2,1)
hier.gt.simulation(N=N,p=p,S=3,psz=psz,Se=Se,Sp=Sp,assayID=Assay)
```

---

mle.prop.eff	*Efficiency of the Proportion Estimator Calculated from Group Testing Data*

Description

This function provides relative efficiency results for the maximum likelihood estimator of disease prevalence (proportion), p. The estimator, $\hat{p}$, is calculated using test responses from commonly used group testing protocols. The relative efficiency values measure how effective group testing is for estimating the prevalence when compared to individual testing. The function also calculates expected testing and estimation costs. Please refer to Warasi and Das (2024), where the statistical methods were introduced.

Usage

```
mle.prop.eff(
  p,
  Se,
  Sp,
  initial.psz,
  protocol = c("MPT", "H2", "H3", "H4", "A2", "A2M"),
  criterion = c("RTE", "REE", "RCE"),
  N = 800,
  ngit = 3000,
  maxit = 200,
  tol = 0.001,
  nrep = 3000,
  seed = NULL,
  ncore = 1
)
```

Arguments

p	Proportion of individuals truly positive for a disease (i.e., disease prevalence).
Se	Assay sensitivity, a scalar value.
Sp	Assay specificity, a scalar value.
initial.psz	A vector of initial pool sizes.
protocol	One of the six group testing protocols, where 'MPT' is the default.

criterion	One of the three optimization criteria, where 'RTE' is the default.
N	Number of individuals to be tested using group testing (i.e., sample size).
ngit	Number of Gibbs iterates used in the EM algorithm.
maxit	Maximum number of iterates in the EM algorithm.
tol	Convergence tolerance for the EM algorithm.
nrep	Number of repetitions used in the proposed computation algorithm.
seed	A single seed value, an integer.
ncore	Number of CPU cores to be used in computing, where ncore => 1.

Details

The function `mle.prop.eff` computes three measures of efficiency: relative testing efficiency (RTE), relative estimation efficiency (REE), and relative cost efficiency (RCE). These measures can be calculated for six common group testing protocols: master pool testing (MPT), hierarchical testing with two, three, and four stages (H2, H3, and H4), and array testing without and with master pool testing (A2 and A2M). For more information on these protocols, refer to Kim et al. (2007). We use the term 'relative efficiency' because these measures compare group testing (numerator) with the usual one-at-a-time, i.e., individual testing (denominator).

In the paper, we defined 'RTE' and discussed how it can be calculated for both common and more complex group testing protocols. For the five multistage protocols (H2, H3, H4, A2, and A2M), our function provides RTE values based on the analytic expressions of Kim et al. (2007). These expressions have been coded by Hitt et al. (2023) in their R package 'binGroup2'. We developed R code to restructure the output obtained from binGroup2, so it is consistent with our proposed method.

Based on the expressions in our article, we analytically calculate REE for MPT and H2 and also compute RCE for MPT. For other scenarios, we determine REE and RCE based on our proposed computation algorithm.

The expected costs, $E[T]$, $E[(\hat{p} - p)^2]$, and $E[T(\hat{p} - p)^2]$, are calculated for a given N, the number of individuals to be tested. The 'MPT' and 'H2' protocols require that N is completely divisible by the initial pool size k. If this is not the case, the integer that is closest to N and divisible by k will be used. It is worth noting that N is also used in the computation algorithm, where a large-sample assumption is made. We found that $N = 800$ may be sufficient in most scenarios for the validity of this assumption, although we used $N = 1200$ in the article; for more information, refer to Warasi and Das (2024).

Arguments 'ngit', 'maxit', and 'tol' are used in the EM algorithm. We found that 'ngit = 3000' Gibbs iterates are generally sufficient, but using a bigger 'ngit' may be more reliable. The default choices for 'maxit' and 'tol' are also reasonable.

Argument 'nrep' is necessary for the validity of the law of large numbers in the computation algorithm. While a larger choice of 'nrep' is generally preferable, our paper suggests that 5000 repetitions may be sufficient in most scenarios. Note that 'nrep = 3000' or so may also provide reasonable approximations.

Execution of 'RCE' for multistage protocols (H2, H3, H4, A2, and A2M) is quite slow because it uses Gibbs samplers in the computation algorithm. For the same reason, execution of 'REE' for H3, H4, A2, and A2M is also slow. To overcome this limitation, we included the `ncore` argument,

which enables users to perform the computing tasks using parallel CPUs through the `parallel` package. The program works with `ncore = 1` or with any larger choice of `ncore`.

Specifying a 'seed' ensures reproducibility of the results produced by our function, whether one uses a single core or multiple cores. Note that the seed value is used only when the computation algorithm is implemented.

Value

A list with three matrix objects labeled as "efficiency", "expected_cost", "iterations":

- **"efficiency"**: A matrix that reports the relative efficiency results: RTE, REE, and RCE.
- **"expected_cost"**: A matrix reporting the expected costs: $E[T]$, $E[(\hat{p}-p)^2]$, and $E[T(\hat{p}-p)^2]$, corresponding to the criteria RTE, REE, and RCE, respectively.
- **"iterations"**: A matrix that provides the cumulative running average of the estimates (sample means) of $E[(\hat{p}-p)^2]$ and $E[T(\hat{p}-p)^2]$ when the computation algorithm is used. These running estimates can be useful for verification of the 'law of large numbers', as demonstrated in the Web Appendix of the article Warasi and Das (2024). In each row of the matrix, the first 1, 2, or 3 columns are for pool size(s) and the remaining 'nrep' columns provide the running averages; see the examples.

These results are provided for all possible pooling configurations within the specified range of initial pool sizes. The stage-specific pool sizes are also included in the results.

References

- Kim HY, Hudgens M, Dreyfuss J, Westreich D, and Pilcher C. (2007). Comparison of Group Testing Algorithms for Case Identification in the Presence of Testing Error. *Biometrics*, 63:1152-1163.
- Zhang W, Liu A, Li Q, Albert P. (2020). Incorporating Retesting Outcomes for Estimation of Disease Prevalence. *Statistics in Medicine*, 39:687-697.
- Warasi and Das (2024). Optimizing Disease Surveillance Through Pooled Testing with Application to Infectious Diseases. *Journal of Agricultural, Biological and Environmental Statistics*. In press.

Examples

```
library(groupTesting)

## Gonorrhea data information:
p0 <- 0.041      # True prevalence
Se <- 0.913      # Assay sensitivity
Sp <- 0.993      # Assay specificity
psz <- 2:6       # A range of initial pool sizes


## Example 1: Two-stage hierarchical testing (H2)

## REE (using closed-form expressions)
res <- mle.prop.eff(p=p0, Se=Se, Sp=Sp, initial.psz=psz, protocol="H2", criterion="REE")
```

```
## Output

# > res
# $efficiency
#       PSZ.S1 PSZ.S2    REE
# [1,]      2      1 0.8920
# [2,]      3      1 0.8932
# [3,]      4      1 0.8975
# [4,]      5      1 0.9034
# [5,]      6      1 0.9104


# $expected_cost
# PSZ.S1 PSZ.S2 E[(phat-p)^2]
# [1,]      2      1  5.731668e-05
# [2,]      3      1  5.732655e-05
# [3,]      4      1  5.767260e-05
# [4,]      5      1  5.805423e-05
# [5,]      6      1  5.864758e-05


# $iterations
# NULL



## RTE (using closed-form expressions)
mle.prop.eff(p=p0, Se=Se, Sp=Sp, initial.psz=psz, protocol="H2", criterion="RTE")

## RCE (using the computation algorithm)
# res <- mle.prop.eff(p=p0, Se=Se, Sp=Sp, initial.psz=psz, protocol="H2", seed=123,
#   criterion="RCE", N=800, ngit=3000, maxit=200, tol=0.001, nrep=3000, ncore=4)

# Note: For 'H2' protocol, get cumulative running averages as:
# res2 <- res$iterations[ ,-(1:2)]
# Now, plot each row; for example: plot(res2[1, ], type='l')
#
# Execution of 'RCE' for H2 is slow, as discussed in the 'details' section.
# The computing challenge can be overcome using multiple CPU cores as shown above.



## Example 2: Three-stage hierarchical testing (H3)

## REE (using the computation algorithm)
# res <- mle.prop.eff(p=p0, Se=Se, Sp=Sp, initial.psz=psz, protocol="H3", seed=123,
#   criterion="REE", N=800, ngit=3000, maxit=200, tol=0.001, nrep=3000, ncore=4)
#
# Note: For 'H3' protocol, get cumulative running averages as:
# res2 <- res$iterations[ ,-(1:3)]  # Now, plot each row: plot(res2[1, ], type='l')

## RTE (using closed-form expressions)
# res <- mle.prop.eff(p=p0, Se=Se, Sp=Sp, initial.psz=psz, protocol="H3",
#   criterion="RTE", N=800, ngit=3000, maxit=200, tol=0.001, nrep=3000, ncore=4)

## RCE (using the computation algorithm)
```

```
# res <- mle.prop.eff(p=p0, Se=Se, Sp=Sp, initial.psz=psz, protocol="H3", seed=123,
#   criterion="RCE", N=800, ngit=3000, maxit=200, tol=0.001, nrep=3000, ncore=4)
```

---

prop.gt   *EM Algorithm to Estimate the Prevalence of a Disease from Group*
                           Testing Data

Description

This function implements an expectation-maximization (EM) algorithm to find the maximum like-
lihood estimate (MLE) of a disease prevalence, p, based on group testing data. The EM algorithm,
which is outlined in Warasi (2023), can model pooling data observed from **any** group testing proto-
col used in practice, including hierarchical and array testing (Kim et al., 2007).

Usage

```
prop.gt(
  p0,
  gtData,
  covariance = FALSE,
  nburn = 2000,
  ngit = 5000,
  maxit = 200,
  tol = 0.001,
  tracing = TRUE,
  conf.level = 0.95
)
```

Arguments

p0	An initial value of the prevalence.
gtData	A matrix or data.frame consisting of the pooled test outcomes and other information from a group testing application. Needs to be specified as shown in the example below.
covariance	When TRUE, the variance is calculated at the MLE.
nburn	The number of initial Gibbs iterates to be discarded.
ngit	The number of Gibbs iterates to be used in the E-step after discarding the initial iterates as a burn-in period.
maxit	The maximum number of EM steps (iterations) allowed in the EM algorithm.
tol	Convergence tolerance used in the EM algorithm.
tracing	When TRUE, progress in the EM algorithm is displayed.
conf.level	Confidence level to be used for the Wald confidence interval.

Details

gtData must be specified as follows. Columns 1-5 consist of the pooled test outcomes (0 for negative and 1 for positive), pool sizes, pool-specific sensitivities, pool-specific specificities, and assay ID numbers, respectively. From column 6 onward, the pool member ID numbers need to be specified. Note that the ID numbers must start with 1 and increase consecutively up to N, the total number of individuals tested. **For smaller pools, incomplete ID numbers must be filled out by -9 or any non-positive numbers** as shown in the example below.

	Z	psz	Se	Sp	Assay	Mem1	Mem2	Mem3	Mem4	Mem5	Mem6
Pool:1	1	6	0.90	0.92	1	1	2	3	4	5	6
Pool:2	0	6	0.90	0.92	1	7	8	9	10	11	12
Pool:3	1	2	0.95	0.96	2	1	2	-9	-9	-9	-9
Pool:4	0	2	0.95	0.96	2	3	4	-9	-9	-9	-9
Pool:5	1	2	0.95	0.96	2	5	6	-9	-9	-9	-9
Pool:6	0	1	0.92	0.90	3	1	-9	-9	-9	-9	-9
Pool:7	1	1	0.92	0.90	3	2	-9	-9	-9	-9	-9
Pool:8	0	1	0.92	0.90	3	5	-9	-9	-9	-9	-9
Pool:9	0	1	0.92	0.90	3	6	-9	-9	-9	-9	-9

This is an example of gtData, where 12 individuals are assigned to 2 non-overlapping initial pools and then tested based on the 3-stage hierarchical protocol. The test outcomes, Z, from 9 pools are in column 1. In three stages, different pool sizes (6, 2, and 1), sensitivities, specificities, and assays are used. The ID numbers of the pool members are shown in columns 6-11. The row names and column names are not required. Note that the EM algorithm can accommodate any group testing data including those described in Kim et al. (2007). For individual testing data, the pool size in column 2 is 1 for all pools.

The EM algorithm implements a Gibbs sampler to approximate quantities required to complete the E-step. Under each EM iteration, ngit Gibbs samples are retained for these purposes after discarding the initial nburn samples.

The variance of the MLE is calculated by an appeal to the missing data principle and the method outlined in Louis (1982).

Value

A list with components:

param	The MLE of the disease prevalence.
covariance	Estimated variance for the disease prevalence.
iterUsed	The number of EM iterations used for convergence.
convergence	0 if the EM algorithm converges successfully and 1 if the iteration limit maxit has been reached.
summary	Estimation summary with Wald confidence interval.

References

Kim HY, Hudgens M, Dreyfuss J, Westreich D, and Pilcher C. (2007). Comparison of Group Testing Algorithms for Case Identification in the Presence of Testing Error. *Biometrics*, 63:1152-1163.

Litvak E, Tu X, and Pagano M. (1994). Screening for the Presence of a Disease by Pooling Sera Samples. *Journal of the American Statistical Association*, 89:424-434.

Liu A, Liu C, Zhang Z, and Albert P. (2012). Optimality of Group Testing in the Presence of Misclassification. *Biometrika*, 99:245-251.

Louis T. (1982). Finding the Observed Information Matrix when Using the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 44:226-233.

Warasi M. (2023). groupTesting: An R Package for Group Testing Estimation. *Communications in Statistics-Simulation and Computation*, 52:6210-6224.

See Also

`hier.gt.simulation` and `array.gt.simulation` for group testing data simulation, and `glm.gt` for group testing regression models.

Examples

```
library(groupTesting)

## To illustrate 'prop.gt', we use data simulated by
## the R functions 'hier.gt.simulation' and 'array.gt.simulation'.

## The simulated data-structures are consistent
## with the data-structure required for 'gtData'.

## Example 1: MLE from 3-stage hierarchical group testing data.
## The data used is simulated by 'hier.gt.simulation'.

N <- 90                # Sample size
S <- 3                 # 3-stage hierarchical testing
psz <- c(6,2,1)        # Pool sizes used in stages 1-3
Se <- c(.95,.95,.98)   # Sensitivities in stages 1-3
Sp <- c(.95,.98,.96)   # Specificities in stages 1-3
assayID <- c(1,2,3)    # Assays used in stages 1-3
p.t <- 0.05            # The TRUE parameter to be estimated

# Simulating data:
set.seed(123)
gtOut <- hier.gt.simulation(N,p.t,S,psz,Se,Sp,assayID)$gtData

# Running the EM algorithm:
pStart <- p.t + 0.2   # Initial value
res <- prop.gt(p0=pStart,gtData=gtOut,covariance=TRUE,
               nburn=2000,ngit=5000,maxit=200,tol=1e-03,
               tracing=TRUE,conf.level=0.95)

# Estimation results:
# > res
```

```
# $param
# [1] 0.05158

# $covariance
#              [,1]
# [1,] 0.0006374296

# $iterUsed
# [1] 4

# $convergence
# [1] 0

# $summary
#      Estimate StdErr 95%lower 95%upper
# prop    0.052  0.025    0.002    0.101

## Example 2: MLE from two-dimensional array testing data.
## The data used is simulated by 'array.gt.simulation'.

N <- 100             # Sample size
protocol <- "A2"     # 2-stage array without testing the initial master pool
n <- 5               # Row/column size
Se <- c(0.95, 0.95)  # Sensitivities
Sp <- c(0.98, 0.98)  # Specificities
assayID <- c(1, 1)   # The same assay in both stages
p.true <- 0.05       # The TRUE parameter to be estimated

# Simulating data:
set.seed(123)
gtOut <- array.gt.simulation(N,p.true,protocol,n,Se,Sp,assayID)$gtData

# Fitting the model:
pStart <- p.true + 0.2  # Initial value
res <- prop.gt(p0=pStart,gtData=gtOut,covariance=TRUE)
print(res)


## Example 3: MLE from non-overlapping initial pooled responses.
## The data used is simulated by 'hier.gt.simulation'.

## Note: With initial pooled responses, our MLE is equivalent
## to the MLE in Litvak et al. (1994) and Liu et al. (2012).

N <- 1000            # Sample size
psz <- 5             # Pool size
S <- 1               # 1-stage testing
Se <- 0.95           # Sensitivity
Sp <- 0.99           # Specificity
assayID <- 1         # Assay used for all pools
p.true <- 0.05       # True parameter
```

```
set.seed(123)
gtOut <- hier.gt.simulation(N,p.true,S,psz,Se,Sp,assayID)$gtData

pStart <- p.true + 0.2   # Initial value
res <- prop.gt(p0=pStart,gtData=gtOut,
               covariance=TRUE,nburn=2000,ngit=5000,
               maxit=200,tol=1e-03,tracing=TRUE)
print(res)


## Example 4: MLE from individual (one-by-one) testing data.
## The data used is simulated by 'hier.gt.simulation'.

N <- 1000           # Sample size
psz <- 1            # Pool size 1 (i.e., individual testing)
S <- 1              # 1-stage testing
Se <- 0.95          # Sensitivity
Sp <- 0.99          # Specificity
assayID <- 1        # Assay used for all pools
p.true <- 0.05      # True parameter

set.seed(123)
gtOut <- hier.gt.simulation(N,p.true,S,psz,Se,Sp,assayID)$gtData

pStart <- p.true + 0.2   # Initial value
res <- prop.gt(p0=pStart,gtData=gtOut,
               covariance=TRUE,nburn=2000,
               ngit=5000,maxit=200,
               tol=1e-03,tracing=TRUE)
print(res)


## Example 5: Using pooled testing data.

# Pooled test outcomes:
Z <- c(1, 0, 1, 0, 1, 0, 1, 0, 0)

# Pool sizes used:
psz <- c(6, 6, 2, 2, 2, 1, 1, 1, 1)

# Pool-specific Se & Sp:
Se <- c(.90, .90, .95, .95, .95, .92, .92, .92, .92)
Sp <- c(.92, .92, .96, .96, .96, .90, .90, .90, .90)

# Assays used:
Assay <- c(1, 1, 2, 2, 2, 3, 3, 3, 3)

# Pool members:
Memb <- rbind(
   c(1, 2,  3,  4,  5,  6),
   c(7, 8,  9, 10, 11, 12),
   c(1, 2, -9, -9, -9, -9),
   c(3, 4, -9, -9, -9, -9),
   c(5, 6, -9, -9, -9, -9),
   c(1,-9, -9, -9, -9, -9),
```

```
    c(2,-9, -9, -9, -9, -9),
    c(5,-9, -9, -9, -9, -9),
    c(6,-9, -9, -9, -9, -9)
)
# The data-structure suited for 'gtData':
gtOut <- cbind(Z, psz, Se, Sp, Assay, Memb)

# Fitting the model:
pStart <- 0.10
res <- prop.gt(p0=pStart,gtData=gtOut,
               covariance=TRUE,nburn=2000,
               ngit=5000,maxit=200,
               tol=1e-03,tracing=TRUE)
print(res)
```

---

waldTest   *Wald Chi-Square Test*

Description

This function implements the Wald *chi-square* test on a Kx1 parameter vector **theta**. The test assumes that **thetaHat**, a consistent estimator of **theta** such as MLE, is asymptotically normal with mean **theta** and covariance matrix **Sigma**. The function can implement 1 test on **theta** as well as multiple, **Q**, tests jointly on **theta**.

Usage

```
waldTest(R, thetaHat, Sigma, r = 0, L = NULL)
```

Arguments

R   A QxK matrix of known coefficients depending on how the test is to be carried out.

thetaHat   An estimate of **theta**.

Sigma   An estimated covariance matrix for thetaHat.

r   A Qx1 matrix of hypothesized values.

L   A character string to be used as a name of the test. When NULL, "L" will be used.

Details

Suppose that Q tests are to be performed jointly on the K by 1 parameter vector **theta**. Let R be a QxK matrix of known coefficients such as 0, 1, and -1, and r be a Qx1 matrix of hypothesized values. The hypotheses are $H0 : R\theta = r$ vs. $H1: R\theta \,!= r$. The test statistic has a chi-square distribution with Q degrees of freedom (Buse, 1982; Agresti, 2002).

Value

A data.frame object of the Wald test results.

References

Agresti A. (2002). Categorical Data Analysis (2nd ed.). Wiley. ISBN 0471360937.

Buse A. (1982). The Likelihood Ratio, Wald, and Lagrange Multiplier Tests: An Expository Note. *The American Statistician*, 36:153-157.

Examples

```
library(groupTesting)

## Example 1
# Parameter: p (proportion)
MLE <- 0.42
Var <- 0.016
# (a) Test  H0: p = 0.50  vs. H1: p != 0.50
R <- matrix(1, nrow=1, ncol=1)
p0 <- 0.50
waldTest( R=R, thetaHat=MLE, r=p0, Sigma=Var )

## Example 2
# Parameter: beta = (beta1, beta2), regression coefficients
MLE <- c(1.09, 2.95)
Cov <- rbind(c(0.21, -0.27),
             c(-0.27, 0.66))
# (a) Test  H0: beta1 = beta2  vs. H1: beta1 != beta2
R <- rbind(c(1,-1))
waldTest( R=R, thetaHat=MLE, r=0, Sigma=Cov, L="1 vs 2" )

# (b) Test  H0: beta1 = 0  vs. H1: beta1 != 0
R <- rbind(c(1,0))
waldTest( R=R, thetaHat=MLE, r=0, Sigma=Cov )

## Example 3
# Parameter: beta = (beta0, beta1, beta2)
MLE <- c(-3.05, 1.99, 0.93)
Cov <- rbind(c( 0.045, -0.022, -0.034),
             c(-0.022,  0.032,  0.008),
             c(-0.034,  0.008,  0.048))

# Performing simultaneous test:
# H0: beta0  = -3, H0: beta1  = 2, H0: beta2  = 1
# H1: beta0 != -3, H1: beta1 != 2, H1: beta2 != 1
R <- rbind(c(1,0,0),
           c(0,1,0),
           c(0,0,1))
r <- matrix( c(-3,2,1), nrow=3 )
waldTest( R=R, thetaHat=MLE, r=r, Sigma=Cov)
```

Index