# Package: cta (via r-universe)

August 25, 2024

**Title** Contingency Table Analysis Based on ML Fitting of MPH Models

**Version** 1.3.0

**Date** 2021-8-21

**Author** Joseph B. Lang [aut], Qiansheng Zhu [aut, cre]

**Maintainer** Qiansheng Zhu <qiansheng-zhu@uiowa.edu>

**Imports** intervals, numDeriv, limSolve, methods

**Suggests** vcd, MASS

**Description** Contingency table analysis is performed based on maximum likelihood (ML) fitting of multinomial-Poisson homogeneous (MPH) and homogeneous linear predictor (HLP) models. See Lang (2004) <doi:10.1214/aos/1079120140> and Lang (2005) <doi:10.1198/016214504000001042> for MPH and HLP models. Objects computed include model goodness-of-fit statistics; likelihood- based (cell- and link-specific) residuals; and cell probability and expected count estimates along with standard errors. This package can also compute test-inversion--e.g. Wald, profile likelihood, score, power-divergence--confidence intervals for contingency table estimands, when table probabilities are potentially subject to equality constraints. For test-inversion intervals, see Lang (2008) <doi:10.1002/sim.3391> and Zhu (2020) <doi:10.17077/etd.005331>.

**License** GPL (>= 2)

**Repository** CRAN

**NeedsCompilation** no

**Date/Publication** 2021-08-23 18:20:10 UTC

## Contents

---

`cta-package` *cta: Contingency Table Analysis Based on ML Fitting of MPH Models*

---

## Description

Contingency table analysis is performed based on maximum likelihood (ML) fitting of multinomial-Poisson homogeneous (MPH) models (Lang, 2004) and homogeneous linear predictor (HLP) models (Lang, 2005). Objects computed include model goodness-of-fit statistics; likelihood-based (cell- and link-specific) residuals; and cell probability and expected count estimates along with standard errors. This package can also compute test-inversion–e.g. Wald, profile likelihood, score, power-divergence–confidence intervals for contingency table estimands, when table probabilities are potentially subject to equality constraints. See Lang (2008) and Zhu (2020) for test-inversion intervals.

## Details

Please call the following two R functions in this **cta** package.

`mph.fit`: Computes maximum likelihood estimates and fit statistics for MPH and HLP models for contingency tables.

`ci.table`: Constructs test-inversion approximate confidence intervals for estimands in contingency tables with or without equality constraints.

## Author(s)

Joseph B. Lang, Qiansheng Zhu

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

Lang, J. B. (2005) Homogeneous linear predictor models for contingency tables, *Journal of the American Statistical Association*, **100**, 121–134.

Lang, J. B. (2008) Score and profile likelihood confidence intervals for contingency table parameters, *Statistics in Medicine*, **27**, 5975–5990.

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

---

`block.fct` *Matrix Direct Sum*

---

## Description

Matrix direct sum function. Creates a block diagonal matrix.

## Usage

```
block.fct(...)
```

## Arguments

| | |
|---|---|
| `...` | R matrices (matrix). |

## Value

`block.fct` returns a block diagonal matrix, where the direct sum is in the order of the input matrices (matrix).

## Author(s)

Joseph B. Lang

## Examples

```
A <- matrix(c(1, 2, 3, 4), nrow = 2, byrow = TRUE)
B <- matrix(c(5, 6, 7, 8, 9, 10), nrow = 2, byrow = TRUE)
C <- matrix(c(11, 12, 13, 14), nrow = 1, byrow = TRUE)
block.fct(A, B, C)
```

---

`check.HLP` *HLP Link Status Check*

---

## Description

Checks whether the link function $L(\cdot)$ is a candidate HLP link function.

Specifically, this program checks whether $L(\cdot)$ satisfies certain necessary conditions that follow from a sufficient condition for HLP link status.

If the necessary conditions are satisfied then there is corroborating evidence that $L(\cdot)$ has HLP link status. If the necessary conditions are not satisfied, then the sufficient condition for HLP link status is not satisfied, so $L(\cdot)$ may or may not have HLP link status.

## Usage

```
check.HLP(L.fct, Z, tol = NULL)
```

## Arguments

| | |
|---|---|
| `L.fct` | An R function object, indicating the link $L(\cdot)$ for HLP link status check. |
| `Z` | Population (aka strata) matrix $Z$. |
| `tol` | The pre-set tolerance with which `norm(diff)` is to be compared with. |

## Details

The main idea:

The model $L(m) = X\beta$ is an HLP model if $L(\cdot)$ is a smooth link function that satisfies the HLP conditions with respect to $Z$ (i.e. strata $s$) and $X$. That is,

- (1) $L(\cdot)$ has HLP link status with respect to $Z$, and
- (2) The implied constraint function $h(m) = U'L(m)$ is $Z$ homogeneous. Here, $null(U') = span(X)$.

Here, (1) $L(\cdot)$ has HLP link status with respect to $Z$ if, for $m = Diag(Z\gamma)p$, equivalently, for $\gamma = Z'm$ and $p = Diag^{-1}(ZZ'm)m$,

- (1)(a) $L(m) = a(\gamma) + L(p)$, where $a(\gamma_1/\gamma_2) - a(1) = a(\gamma_1) - a(\gamma_2)$, i.e. $a(\gamma)$ has the form $C \log \gamma + \texttt{constant}$; or
- (1)(b) $L(m) = G(\gamma)L(p)$, where $G(\gamma)$ is a diagonal matrix with diagonal elements that are powers of the $\gamma$ elements, i.e. $L(\cdot)$ is $Z$ homogeneous (see Lang (2004)); or

- (1)(c) The components of $L(\cdot)$ are a mixture of types (a) and (b): $L_j(m) = a_j(\gamma) + L_j(p)$ or $L_j(m) = G_j(\gamma)L_j(p)$, $j = 1, \ldots, l$.

N.B. Lang (2005) defined HLP models as those satisfying (1)(a) and (2). `mph.fit` uses a broader definition of HLP model. Specifically, models satisfying (1)(b) and (2) or (1)(c) and (2) are also considered HLP models.

Conditions (1)(b) and (2) can be checked using the `check.homog` function. Condition (1)(c) is not checked.

This function, `check.HLP`, is concerned with sufficient condition (1)(a) only. If $L(\cdot)$ satisfies (1)(a) then

- (i) `diff1` $= [L(Diag(Z\gamma_1)p_1) - L(Diag(Z\gamma_2)p_1)] - [L(Diag(Z\gamma_1/\gamma_2)p_1) - L(p_1)] = 0$, and
- (ii) `diff2` $= [L(Diag(Z\gamma_1)p_1) - L(Diag(Z\gamma_1)p_2)] - [L(p_1) - L(p_2)] = 0$.

Here $p_i = Diag^{-1}(ZZ'm_i)m_i$, where $m_i = Diag(Z\gamma_i)p_i$, $i = 1, 2$.

This program randomly generates `g1` ($\gamma_1$), `g2` ($\gamma_2$), `p1`, `p2`, and computes `norm(diff) = sqrt(norm(diff1)^2 + norm(diff2)^2)`. It returns a warning if `norm(diff)` is too far from 0.

## Value

`check.HLP` returns a character string chk. If chk = `""`, then there is corroborating evidence that $L(\cdot)$ has HLP link status. If chk = `paste("L(m) may not be an HLP link [based on tol=",tol,"]!")`, then the sufficient condition for HLP link status is not satisfied, so $L(\cdot)$ may or may not have HLP link status.

## Author(s)

Joseph B. Lang

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

Lang, J. B. (2005) Homogeneous linear predictor models for contingency tables, *Journal of the American Statistical Association*, **100**, 121–134.

## See Also

`mph.fit`, `check.homog`, `check.zero.order.homog`

## Examples

```
# 3-by-3-by-3 Table.
# For a description of the model, see Michael Haber's Example 2,
# p. 433, in  Biometrics (in Shorter Communications), Vol. 42,
# No. 2. (Jun., 1986), pp. 429-435.
A <- gl(3, 9, 27)
B <- gl(3, 3, 27)
C <- gl(3, 1, 27)
```

```
MAB <- kronecker(diag(9), matrix(1, 1, 3))
MAC <- kronecker(diag(3), kronecker(matrix(1, 1, 3), diag(3)))
MBC <- kronecker(matrix(1, 1, 3), diag(9))
M <- rbind(MAB, MAC, MBC)
Mr <- M[-c(3, 6, 7, 8, 9, 12, 15, 16, 17, 18, 21, 24,
           25, 26, 27), ]
C <- c(1, -1, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, -1, -1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, -1, -1, 1)
C <- matrix(C, 3, 12, byrow = TRUE)
L.fct <- function(m) {
  p <- m / sum(m)
  C %*% log(Mr %*% p)
}
Z <- matrix(rep(1, 27), ncol = 1)
check.HLP(L.fct, Z)
```

## check.homog *Z Homogeneity Check*

### Description

Checks whether the constraint function $h(\cdot)$ satisfies a necessary condition for $Z$ homogeneity.

### Usage

```
check.homog(h.fct, Z, tol = NULL)
```

### Arguments

| | |
|---|---|
| h.fct | An R function object, indicating the constraint function $h(\cdot)$ for $Z$ homogeneity check. |
| Z | Population (aka strata) matrix $Z$. |
| tol | The pre-set tolerance with which `norm(diff)` is to be compared with. |

### Details

The main idea:

$h(\cdot)$ is $Z$ homogeneous if $h(Diag(Z\gamma)x) = G(\gamma)h(x)$, where $G$ is a diagonal matrix with $\gamma$ elements raised to some power.

As a check, if $h(\cdot)$ is homogeneous then

$$h(Diag(Z\gamma)x_1)/h(Diag(Z\gamma)x_2) = h(x_1)/h(x_2);$$

That is,

$$\texttt{diff} = h(Diag(Z\gamma)x_1)h(x_2) - h(Diag(Z\gamma)x_2)h(x_1) = 0.$$

Here, the division and multiplication are taken element-wise.

This program randomly generates `gamma`, `x1`, and `x2`, and computes `norm(diff)`. It returns a warning if `norm(diff)` is too far from 0.

## Value

`check.homog` returns a character string `chk` that states whether $h(\cdot)$ is $Z$ homogeneous. If `chk` = `""`, it means that based on the necessary condition, we cannot state that $h(\cdot)$ is not $Z$ homogeneous.

## Author(s)

Joseph B. Lang

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

## See Also

`check.zero.order.homog`, `mph.fit`, `check.HLP`

## Examples

```
# EXAMPLE 1
h.fct <- function(m) {m[1] - m[2]}
Z <- matrix(c(1, 1), nrow = 2)
check.homog(h.fct, Z)

# EXAMPLE 2
h.fct.2 <- function(m) {m[1]^2 - m[2]}
Z <- matrix(c(1, 1), nrow = 2)
check.homog(h.fct.2, Z)
```

---

`check.zero.order.homog` *Zero-Order $Z$ Homogeneity Check*

---

## Description

Checks whether the estimand function $S(\cdot)$ is zero-order $Z$ homogeneous.

## Usage

```
check.zero.order.homog(S.fct, Z, tol = 1e-9)
```

## Arguments

| | |
|---|---|
| `S.fct` | An R function object, indicating the estimand function $S(\cdot)$ for zero-order $Z$ homogeneity check. |
| `Z` | Population (aka strata) matrix $Z$. |
| `tol` | The pre-set tolerance with which `norm(diff.LRHS)` is to be compared with. |

## Details

The main idea:

$S(\cdot)$ is zero-order $Z$ homogeneous if $S(Diag(Z\gamma)x) = S(x)$, for all $\gamma > 0$, and for all $x$ within its domain. This program randomly generates `gam` ($\gamma$) and `x` ($x$), and computes

$$\texttt{diff.LRHS} = S(Diag(Z\gamma)x) - S(x).$$

It returns a warning if `norm(diff.LRHS)` is too far from $0$.

## Value

`check.zero.order.homog` returns a character string `check.result` that states whether $S(\cdot)$ is zero-order $Z$ homogeneous. If `check.result = ""`, it means that we cannot state that $S(\cdot)$ is not zero-order $Z$ homogeneous based on the result of the check.

## Author(s)

Qiansheng Zhu

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

## See Also

`check.homog`, `check.HLP`

## Examples

```
# EXAMPLE 1
S.fct <- function(m) {(m[1] - m[2]) / (m[1] + m[2])}
Z <- matrix(c(1, 1, 1, 1), nrow = 4)
check.zero.order.homog(S.fct, Z)

# EXAMPLE 2
S.fct.2 <- function(m) {m[1] - m[2]}
Z <- matrix(c(1, 1, 1, 1), nrow = 4)
check.zero.order.homog(S.fct.2, Z)
```

---

# `ci.table` *Test-Inversion CIs for Estimands in Contingency Tables*

---

## Description

Constructs test-inversion approximate confidence intervals (CIs) for estimands in contingency tables subject to equality constraints. Test statistics include Wald-type statistics, and difference and nested versions of power-divergence statistics. This program can also compute test-inversion approximate confidence intervals for estimands in contingency tables without additionally imposed equality constraints, by setting the constraint function `h.fct = 0`.

## Usage

```
ci.table(y, h.fct = 0, h.mean = FALSE, S.fct, S.mean = FALSE, S.P = FALSE,
         S.space.H0 = NULL, method = "all", cc = 0.95, pdlambda = 2/3,
         trans.g = NULL, trans.g.epsilon = 0, trans.g.inv = NULL,
         strata = rep(1, length(y)), fixed.strata = "all", delta = 0.5,
         max.iter = 50, tol = 1e-2, tol.psi = 1e-4, adj.epsilon = 0.03,
         iter.robust.max = 30, iter.robust.eff = 10, check.homog.tol = 1e-9,
         check.zero.order.homog.tol = 1e-9, max.mph.iter = 1000, step = 1,
         change.step.after = 0.25 * max.mph.iter, y.eps = 0, iter.orig = 5,
         norm.diff.conv = 1e-6, norm.score.conv = 1e-6,
         max.score.diff.iter = 10, h0.fct.deriv = NULL,
         S0.fct.deriv = NULL, trans.g.deriv = NULL, plot.CIs = TRUE)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `h.fct` | The imposed equality constraint(s). Note that sampling constraints are not included in `h.fct`, and the imposed equality constraints should be non-redundant.<br>If `h.mean = FALSE` (default), $h(p)$ should be the input, where $p$ is the vector of data model probabilities, or it can be called the vector of table probabilities; If `h.mean = TRUE`, $h(m)$ should be the input, where $m$ is the vector of expected table counts, i.e. $m = E(Y)$. In the case of $h(m)$ being the input, the function $h(\cdot)$ should be $Z$ homogeneous, where $Z$ is the population matrix. For the definition of $Z$ homogeneity and the population matrix, see Lang (2004). Note that if there is no imposed equality constraint, we should input `h.fct = 0` (real number `0`). Please do not specify `h.fct` as a zero function in this case. On the contrary, if there is (are) imposed equality constraint(s), please specify `h.fct` as an R function. Another important note is that if there are multiple imposed equality constraints, please use `rbind()`, not `c()`, to concatenate the imposed equality constraints into a column vector.<br>By default, `h.fct = 0`. |
| `h.mean` | Logical argument, `TRUE` or `FALSE`. If `h.mean = FALSE` (default), the input `h.fct` is treated as a function of $p$; If `h.mean = TRUE`, the input `h.fct` is treated as a function of $m$. |
| `S.fct` | Parameter or estimand of interest. It should be an R function, which returns a real number. i.e. $S(\cdot)$ is a real-valued function. If `S.mean = FALSE` and `S.P = FALSE` (default), $S(p)$ should be the input; If `S.mean = TRUE`, $S(m)$ should be the input; If `S.P = TRUE`, $S(P)$ should be the input, where $P$ is the vector of joint probabilities, or it can be called the vector of pre-data probabilities. In the case of $S(m)$ or $S(P)$ being the input, the function $S(\cdot)$ should be zero-order $Z$ homogeneous, then $S(P)$ is $Z$ estimable with $S(P) = S(m)$. In addition, when we are in the process of computing test-inversion confidence intervals other than Wald intervals, we have to fit several models and obtain constrained MLEs of expected table counts. These models have equality constraints $h_0^*(m) = 0$, where $h_0^*(m) = (h_0'(m), S_0(m) - \psi, samp_0'(m))'$. Here $h_0(m) = 0$ is (are) the imposed equality constraint(s), written in terms of $m$; $S_0(m) - \psi = 0$ means that the estimand of interest is equal to $\psi$; $samp_0(m) = 0$ is (are) the sampling |

constraint(s), written in terms of $m$. Restriction of $S(m)$ [or $S(P)$] to zero-order $Z$ homogeneity guarantees the $Z$ homogeneity of $h_0^*(m)$.

`S.mean, S.P` Logical argument, `TRUE` or `FALSE`. If `S.mean = FALSE` and `S.P = FALSE` (default), the input `S.fct` is treated as a function of $p$; If `S.mean = TRUE`, the input `S.fct` is treated as a function of $m$; If `S.P = TRUE`, the input `S.fct` is treated as a function of $P$.

`S.space.H0` Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. If `S.space.H0` is not specified or the input `S.space.H0 = NULL`, the restricted estimand space is treated as $(-\infty, \infty)$, i.e. the whole real number line. If `S.space.H0` is specified, it can either be input as a vector of length of an even number, or be input in class `Intervals_full {intervals}`. As an example, if the restricted estimand space is $(-\infty, -1] \cup [1, \infty)$, then the input `S.space.H0` could be `c(-Inf, -1, 1, Inf)`, or `Intervals_full(matrix(c(-Inf, -1, 1, Inf), ncol = 2, byrow = TRUE), closed = matrix(c(FALSE, TRUE, TRUE, FALSE), ncol = 2, byrow = TRUE), type = "R")`. It is strongly recommended that `S.space.H0` be specified, as it will improve the accuracy and (possibly) speed in interval estimation. However, it is often difficult to have an idea of the restricted estimand space exactly. In this scenario, specification of one (or several) possibly larger interval(s) that cover(s) the exact restricted estimand space is also helpful.

`method` The test statistic(s) in constructing the test-inversion approximate confidence interval(s). There are eight different test statistics, and the user is allowed to choose any number of the test statistics out of the eight. The eight test statistics are listed as follows: `"Wald"`, `"trans.Wald"` (need specification of the transformation $g$), `"diff.Xsq"`, `"nested.Xsq"`, `"diff.Gsq"` (same as `"PL"` or `"LR"`), `"nested.Gsq"`, `"diff.PD"`, `"nested.PD"` (need specification of the power-divergence index parameter $\lambda$). If the input `method = "all"` (default), all test statistics will be employed to compute confidence intervals.

`cc` Confidence coefficient, or the nominal level of the confidence interval.

`pdlambda` The index parameter $\lambda$ in the power-divergence statistic.

`trans.g` The transformation $g$ used in the transformed Wald confidence interval. First, we construct a confidence interval for $g(S(\cdot))$, then we back-transform, i.e. apply $g^{-1}$ to the endpoints in order to obtain a confidence interval for $S(\cdot)$. There are several built-in options for the transformation: `"Fisher's z"`, `"log"`, `"-log"` (same as `"negative log"`), and `"[A, B]"`. `"[A, B]"` refers to the reparameterization trick as stated in the Discussion part of Lang (2008). The user is also allowed to input their own choice of `trans.g`. Ordinarily, the transformation $g$ should be a bijection. Ideally, $g$ should be smooth, strictly monotonically increasing, and "to parameterize away the boundary" (Lang, 2008).

`trans.g.epsilon`
The small $\epsilon$ adjustment included in the transformation $g$. For example, the `"[A, B]"` transformation $g$ with the small $\epsilon$ is

$$g(x) = \log(x - A + \epsilon) - \log(B + \epsilon - x).$$

By default, `trans.g.epsilon = 0`.

| | |
|---|---|
| `trans.g.inv` | $g^{-1}$ function used in back-transformation step in construction of the transformed Wald confidence interval. If `trans.g` is any one of the built-in options, then `trans.g.inv` is automatically specified accordingly. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. By default, `strata = rep(1, length(y))` refers to the single stratum (non-stratified) setting. As another example, `strata = c(1,1,2,2)` means that the first and second table cells belong to the first stratum, and the third and fourth table cells belong to the second stratum. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. It can equal one of the keywords, `fixed.strata = "all"` or `fixed.strata = "none"`, or it can be a vector of stratum membership identifiers, e.g. `fixed.strata = c(1,3)` or `fixed.strata = c("pop1", "pop5")`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. By default, `delta = 0.5`. |
| `max.iter` | One of the stopping criteria. It is the maximum number of iterations in the sliding quadratic root-finding algorithm for searching the two roots to the test-inversion equation. |
| `tol` | One of the stopping criteria. In solving for the roots to the test-inversion equation, if the test statistic for testing $H_0(\psi) : S_0(m) = \psi$ vs. not $H_0(\psi)$ under the general hypothesis $H_0 : (h_0'(m), samp_0'(m))' = 0$, for a certain $\psi$, is within `tol` of the critical value, then we stop the iterations, and this current $\psi$ is treated as one root. Note that since we are constructing approximate (contrary to exact) confidence intervals based on the asymptotic distribution under the null hypothesis, `tol` need not be too small. |
| `tol.psi` | One of the stopping criteria. In solving for the roots to the test-inversion equation, if the two $\psi$'s that are in nearby iterates in the corresponding tests $H_0(\psi)$ vs. not $H_0(\psi)$ under the general hypothesis $H_0$, are less than `tol.psi` apart in distance, then we stop the iterations, and the current $\psi$ is treated as one root. Note that we should specify `tol.psi` to be sufficiently small (compared with the size of the restricted estimand space) so that the iterations are to be terminated mainly because of closeness of the test statistic to the critical value. |

`adj.epsilon`, `iter.robust.max`, `iter.robust.eff`

The parameters used in the robustifying procedure. First, we attempt to construct confidence intervals based on the original data `y`, but an error might occur during this process. The reason for occurrence of the error might be the non-existence of the constrained MLE subject to $H_0$, or it might be because of the fact that the $\psi$ in the hypothesis test $H_0(\psi)$ vs. not $H_0(\psi)$ is, on some scale, too far away from $\widehat{\psi}$ which is the constrained MLE of the estimand subject to $H_0$, although this $\psi$ is still within the restricted estimand space. If an error, or non-convergence issue occurs, then the program will go through the robustifying procedure, with the goal of reporting a confidence interval anyway, even in the most extreme configuration and/or with the most "extreme" data.

In the robustifying procedure, we adjust the original data `y` by adding `1 * adj.epsilon` to each original table count, and compute the confidence interval based on the adjusted data `y + 1 * adj.epsilon`. Note, however, that even the adjusted data may lead to non-convergence issue sometimes. We also adjust the original data by adding `2 * adj.epsilon`, ..., `iter.robust.max * adj.epsilon`, and compute confidence intervals based on these adjusted data, respectively. For computing purposes, as soon as `iter.robust.eff` confidence intervals based on the adjusted data have been successfully computed, we will not proceed further into adjustment and interval estimation based on adjusted data. Now, by exploiting the property that

$$\lim_{\texttt{adj.epsilon} \to 0+} CI(y + \texttt{adj.epsilon}; H_0) = CI(y; H_0),$$

we extrapolate using a polynomial fit of degree at most three based on lower and upper endpoints of the confidence intervals on adjusted data. It is advised that `adj.epsilon` should not exceed `0.1`, but it should not be too small. By default, `adj.epsilon = 0.03`.

`check.homog.tol`
: Round-off tolerance for $Z$ homogeneity check. If the function $h(\cdot)$ with respect to $m$ is not $Z$ homogeneous, the algorithm will stop immediately and report an error.

`check.zero.order.homog.tol`
: Round-off tolerance for zero-order $Z$ homogeneity check. If the function $S(\cdot)$ with respect to $m$ or $P$ is not zero-order $Z$ homogeneous, the algorithm will stop immediately and report an error.

`max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter`
: The parameters used in `mph.fit`.

`h0.fct.deriv`
: The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. In this algorithm, if the input function `h.fct` is a function of $p$, then the algorithm automatically rewrites it into another function of $m$: $h(p) = h(Diag^{-1}(ZZ'm)m) = h_0(m)$. If the input function `h.fct` is a function of $m$, then we let $h_0(m) = h(m)$. `h0.fct.deriv`, if it is specified, equals $\partial h_0'(m)/\partial m$. Note that if $h_0(\cdot)$ maps from $R^p$ to $R^q$, i.e. there are $q$ constraints, then `h0.fct.deriv` returns a $p$-by-$q$ matrix of partial derivatives. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used.

`S0.fct.deriv`
: The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. In this algorithm, if the input function `S.fct` is a function of $p$, then the algorithm automatically rewrites it into another function of $m$: $S(p) = S(Diag^{-1}(ZZ'm)m) = S_0(m)$. If the input function `S.fct` is a function of $m$, then we let $S_0(m) = S(m)$. If the input function `S.fct` is a function of $P$, since $S(\cdot)$ is required to be zero-order $Z$ homogeneous, in which case $S(P) = S(m)$, we let $S_0(m) = S(P)$. `S0.fct.deriv`, if it is specified, equals $\partial S_0(m)/\partial m$. It is a column vector, whose length is the same as the length of $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used.

`trans.g.deriv`
: The derivative function of the transformation $g$, i.e. $dg(w)/dw$. If it is specified, it should be an R function, even if the derivative function is a constant function.

`plot.CIs`
: Logical argument, `TRUE` or `FALSE`. If `plot.CIs = TRUE` (default), a visual display of the computed confidence interval(s) will be created. If `plot.CIs = FALSE`, no plots will be created.

## Value

`ci.table` returns a list, which includes the following objects:

| | |
|---|---|
| `result.table` | A table that displays lower and upper endpoints of the computed confidence interval(s). The length(s) of the confidence interval(s) is (are) reported in the last column. |
| `CIs` | An object of class `Intervals_full {intervals}` that includes all of the computed confidence interval(s). |
| `Shat` | The constrained MLE of $S(\cdot)$ subject to $H_0$. If there is an error or non-convergence issue during the process of fitting the model subject to $H_0$ by `mph.fit`, `Shat` is set to be `NA`; or if the constrained MLE does not exist, `Shat` is also set to be `NA`. |
| `ase.Shat` | The asymptotic standard error, i.e. ase, of the constrained MLE of $S(\cdot)$ subject to $H_0$. If there is an error or non-convergence issue during the process of fitting the model subject to $H_0$ by `mph.fit`, `ase.Shat` is set to be `NA`; or if the constrained MLE does not exist, `ase.Shat` is also set to be `NA`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$. It might be different from the input `S.space.H0`. If the input `S.space.H0` is the union of at least two disjoint intervals, then the output `S.space.H0` displays the particular interval in which `Shat`, the constrained MLE of $S(\cdot)$ subject to $H_0$, lies. If the input `S.space.H0` is an interval, then the output `S.space.H0` is the same as the input. If `S.space.H0` is unspecified or `S.space.H0 = NULL` in the input, then the output `S.space.H0 = NULL`. |
| `cc` | Confidence coefficient, or the nominal level of the confidence interval. It is the same as the `cc` in the input. |
| `method` | The test statistic(s) that is (are) actually used to construct the test-inversion approximate confidence interval(s). |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. It is the same as the `pdlambda` in the input. |
| `warnings.collection` | Includes all of the warning messages that occur during construction of the confidence interval(s). They might be on evoking of the robustifying procedure: `"xxx.CI: Adjustment used. Not on original data.\n"`, or they might be on unsuccessful construction of the confidence interval(s): `"xxx.CI: NA.\n"` |

## Author(s)

Qiansheng Zhu

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

Lang, J. B. (2008) Score and profile likelihood confidence intervals for contingency table parameters, *Statistics in Medicine*, **27**, 5975–5990.

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

mph.fit, mph.summary

## Examples

```
### Construct test-inversion CIs subject to equality constraints.

# I. Mice-Fungicide data: Innes et al. (1969) conducted an experiment
#    to test the possible carcinogenic effect of a fungicide Avadex on
#    four subgroups of mice. The data is reproduced as a 2-by-2-by-4
#    three-way contingency table. Within each of the four 2-by-2 two-way
#    sub-tables, there is one fixed stratum for the treated group, and
#    there is also one fixed stratum for the control group. Overall,
#    the data was collected under the product-multinomial sampling scheme.
#    We assume that the relative risks that correspond to the four 2-by-2
#    two-way sub-tables are the same, and we construct 95% test-inversion
#    confidence intervals for this common relative risk.
#
#    For a detailed description of the Mice-Fungicide data set, see
#    Gart (1971):
#    Gart, J. J. (1971) The comparison of proportions: a review of
#    significance tests, confidence intervals and adjustments for
#    stratification. Revue de l'Institut International de Statistique,
#    39(2), pp. 148-169.

obs.y <- c(4, 12, 5, 74, 2, 14, 3, 84, 4, 14, 10, 80, 1, 14, 3, 79)

h.fct <- function(p) {
  RR_1 <- p[1] / p[3]
  RR_2 <- p[5] / p[7]
  RR_3 <- p[9] / p[11]
  RR_4 <- p[13] / p[15]
  rbind(RR_1 - RR_2, RR_1 - RR_3, RR_1 - RR_4)
}

S.fct <- function(p) {
  p[1] / p[3]
}

mice_result <- ci.table(obs.y, h.fct = h.fct, S.fct = S.fct,
                        S.space.H0 = c(0, Inf), trans.g = "log",
                        strata = rep(seq(1, 8), each = 2))


# II. Suppose there is a 3-by-4-by-2 three-way contingency table which
#     cross-classifies three variables: X, Y, and Z. We assign scores
#     {1,2,3}, {1,2,3,4}, and {1,2} to the variables X, Y, and Z,
#     respectively. At each level of Z, there is a 3-by-4 two-way sub-table
#     for variables X and Y, and the 3-by-4 sub-table forms a fixed
#     stratum. We assume that the Pearson's correlation coefficient between
#     X and Y when Z = 1 is the same as that when Z = 2. The observed table
#     counts are (1,2,3,4,5,6,7,8,9,10,11,12) for the 3-by-4 sub-table when
#     Z = 1, and (13,14,15,16,17,18,19,20,21,22,23,24) for the 3-by-4 sub-
#     table when Z = 2. We construct a 95% profile likelihood confidence
#     interval for this common Pearson's correlation coefficient.

corr_freq_prob <- function(freq, score.X, score.Y) {
  # Compute the Pearson's correlation coefficient based on the vector
  # of table (frequency) counts or the vector of underlying table
  # probabilities.
  # Note that the input freq is a vector.
  c <- length(score.X)
  d <- length(score.Y)
  freq <- matrix(freq, nrow = c, ncol = d, byrow = TRUE)
  P <- freq / sum(freq)
  P.row.sum <- apply(P, 1, sum)
  P.column.sum <- apply(P, 2, sum)
  EX <- crossprod(score.X, P.row.sum)
  EY <- crossprod(score.Y, P.column.sum)
  EXsq <- crossprod(score.X^2, P.row.sum)
  EYsq <- crossprod(score.Y^2, P.column.sum)
  sdX <- sqrt(EXsq - EX^2)
  sdY <- sqrt(EYsq - EY^2)
  EXY <- 0
  for (i in seq(1, c)) {
    for (j in seq(1, d)) {
      EXY <- EXY + score.X[i] * score.Y[j] * P[i, j]
    }
  }
  Cov.X.Y <- EXY - EX * EY
  if (Cov.X.Y == 0) {
    corr <- 0
  }
  else {
    corr <- as.numeric(Cov.X.Y / (sdX * sdY))
  }
  corr
}

h.fct <- function(p) {
  corr_1 <- corr_freq_prob(p[seq(1, 12)], c(1, 2, 3), c(1, 2, 3, 4))
  corr_2 <- corr_freq_prob(p[seq(13, 24)], c(1, 2, 3), c(1, 2, 3, 4))
  corr_1 - corr_2
}

S.fct <- function(p) {
  corr_freq_prob(p[seq(1, 12)], c(1, 2, 3), c(1, 2, 3, 4))
}

corr_result <- ci.table(y = seq(1, 24), h.fct = h.fct, S.fct = S.fct,
                        S.space.H0 = c(-1, 1), method = "LR",
                        trans.g = "Fisher's z", strata = rep(c(1, 2), each = 12),
                        plot.CIs = FALSE)
```

```
# III. Crying Baby data: Gordon and Foss (1966) conducted an experiment to
#      investigate the effect of rocking on the crying of full term babies.
#      The data set can be reproduced as a 2-by-2-by-18 three-way contingency
#      table. Within each of the eighteen 2-by-2 two-way sub-tables, there is
#      one fixed stratum for the experimental group and one fixed stratum for
#      the control group. Overall, the data was collected under the product-
#      multinomial sampling scheme. We assume common odds ratios among the
#      eighteen two-way sub-tables, and we construct 95% test-inversion
#      confidence intervals for this common odds ratio.
#
#      For a detailed description of the Crying Baby data set, see Cox (1966):
#      Cox, D. R. (1966) A simple example of a comparison involving quantal
#      data. Biometrika, 53(1-2), pp. 213-220.

obs.y <- c(0,1,5,3,0,1,4,2,0,1,4,1,1,0,5,1,0,1,1,4,0,1,5,4,0,1,3,5,0,1,
           4,4,0,1,2,3,1,0,1,8,0,1,1,5,0,1,1,8,0,1,3,5,0,1,1,4,0,1,2,4,
           0,1,1,7,1,0,2,4,0,1,3,5)
strata <- rep(seq(1, 36), each = 2)

h.fct <- function(p) {
  OR_1 <- p[1] * p[4] / (p[2] * p[3])
  OR_2 <- p[5] * p[8] / (p[6] * p[7])
  OR_3 <- p[9] * p[12] / (p[10] * p[11])
  OR_4 <- p[13] * p[16] / (p[14] * p[15])
  OR_5 <- p[17] * p[20] / (p[18] * p[19])
  OR_6 <- p[21] * p[24] / (p[22] * p[23])
  OR_7 <- p[25] * p[28] / (p[26] * p[27])
  OR_8 <- p[29] * p[32] / (p[30] * p[31])
  OR_9 <- p[33] * p[36] / (p[34] * p[35])
  OR_10 <- p[37] * p[40] / (p[38] * p[39])
  OR_11 <- p[41] * p[44] / (p[42] * p[43])
  OR_12 <- p[45] * p[48] / (p[46] * p[47])
  OR_13 <- p[49] * p[52] / (p[50] * p[51])
  OR_14 <- p[53] * p[56] / (p[54] * p[55])
  OR_15 <- p[57] * p[60] / (p[58] * p[59])
  OR_16 <- p[61] * p[64] / (p[62] * p[63])
  OR_17 <- p[65] * p[68] / (p[66] * p[67])
  OR_18 <- p[69] * p[72] / (p[70] * p[71])
  rbind(OR_1 - OR_2, OR_1 - OR_3, OR_1 - OR_4, OR_1 - OR_5, OR_1 - OR_6,
        OR_1 - OR_7, OR_1 - OR_8, OR_1 - OR_9, OR_1 - OR_10, OR_1 - OR_11,
        OR_1 - OR_12, OR_1 - OR_13, OR_1 - OR_14, OR_1 - OR_15,
        OR_1 - OR_16, OR_1 - OR_17, OR_1 - OR_18)
}

S.fct <- function(p) {
  p[1] * p[4] / (p[2] * p[3])
}

crying_baby_result <- ci.table(obs.y, h.fct = h.fct, S.fct = S.fct,
                               S.space.H0 = c(0, Inf), trans.g = "log",
                               strata = strata, fixed.strata = "all",
                               y.eps = 0.4)
```

```
# IV. Homicide data: Radelet & Pierce (1985) examined cases of 1017 homicide
#     defendants in Florida between 1973 and 1977. Both the police department
#     and prosecutors classified these cases into three mutually exclusive
#     categories: 1 = "No Felony", 2 = "Possible Felony", 3 = "Felony".
#     Three variables: police classification (P), court (i.e. prosecutors')
#     classification (C), and race of defendant/victim (R) are cross-
#     classified in a 3-by-3-by-4 three-way contingency table. The data
#     was collected based on independent Poisson sampling, and the strata
#     correspond to levels of the race combination (R).
#
#     For a detailed description of the Homicide data set, see Agresti (1984)
#     and Radelet & Pierce (1985):
#     Agresti, A. (1984). Analysis of Ordinal Categorical Data. John Wiley &
#     Sons.
#     Radelet, M. L., & Pierce, G. L. (1985). Race and prosecutorial
#     discretion in homicide cases. Law & Society Review, 19(4), pp. 587-622.
#
#     To measure agreement between police and court classifications, the four
#     estimands of interest are Cohen's unweighted kappa coefficients at four
#     levels of R, respectively. We construct 95% test-inversion confidence
#     intervals for the estimands subject to two sets of equality constraints,
#     respectively.
#     (1) WkW and BkB have the same unweighted kappa, and BkW and WkB have
#     the same unweighted kappa.
#     (2) A "row effects" model for the conditional R-C association:
#     log mu_{ijk} = lambda + lambda_{i}^{R} + lambda_{j}^{P} + lambda_{k}^{C} +
#           lambda_{ij}^{RP} + lambda_{jk}^{PC} + tau_{i}^{RC}(w_{k} - bar{w}),
#     where race effects {tau_{i}^{RC}} that sum to zero are introduced for an
#     R-C association. The variable C is viewed as being ordinal with integer
#     monotonic scores {w_{k}}={1,2,3}.

BkW_v <- c(7, 1, 3, 0, 2, 6, 5, 5, 109)
WkW_v <- c(236, 11, 26, 7, 2, 21, 25, 4, 101)
BkB_v <- c(328, 6, 13, 7, 2, 3, 21, 1, 36)
WkB_v <- c(14, 1, 0, 6, 1, 1, 1, 0, 5)
obs.y <- c(BkW_v, WkW_v, BkB_v, WkB_v)

Unweighted.Kappa.BkW <- function(p) {
  mat.p <- matrix(p[seq(1,9)], nrow = 3, byrow = TRUE)
  Kappa(mat.p)$Unweighted[1]
}
Unweighted.Kappa.WkW <- function(p) {
  mat.p <- matrix(p[seq(10,18)], nrow = 3, byrow = TRUE)
  Kappa(mat.p)$Unweighted[1]
}
Unweighted.Kappa.BkB <- function(p) {
  mat.p <- matrix(p[seq(19,27)], nrow = 3, byrow = TRUE)
  Kappa(mat.p)$Unweighted[1]
}
Unweighted.Kappa.WkB <- function(p) {
  mat.p <- matrix(p[seq(28,36)], nrow = 3, byrow = TRUE)
  Kappa(mat.p)$Unweighted[1]
```

```
}

# Constraints (1)
library(vcd)
WkW.BkB_BkW.WkB_cons <- function(p) {
  mat.BkW <- matrix(p[seq(1,9)], nrow = 3, byrow = TRUE)
  mat.WkW <- matrix(p[seq(10,18)], nrow = 3, byrow = TRUE)
  mat.BkB <- matrix(p[seq(19,27)], nrow = 3, byrow = TRUE)
  mat.WkB <- matrix(p[seq(28,36)], nrow = 3, byrow = TRUE)
  rbind(Kappa(mat.BkW)$Unweighted[1] - Kappa(mat.WkB)$Unweighted[1],
        Kappa(mat.WkW)$Unweighted[1] - Kappa(mat.BkB)$Unweighted[1])
}
homicide_kappa_same_fit <- mph.fit(obs.y, h.fct = WkW.BkB_BkW.WkB_cons,
                                   strata = rep(c(1,2,3,4), each = 9),
                                   fixed.strata = "none")
homicide_kappa_same_fit$Gsq
pchisq(homicide_kappa_same_fit$Gsq, 2, lower.tail = FALSE)  # p-value

BkW_kappa_same <- ci.table(obs.y, h.fct = WkW.BkB_BkW.WkB_cons,
                           S.fct = Unweighted.Kappa.BkW, S.space.H0 = c(0,1),
                           strata = rep(c(1,2,3,4), each = 9),
                           fixed.strata = "none", trans.g = "[A,B]")
WkW_kappa_same <- ci.table(obs.y, h.fct = WkW.BkB_BkW.WkB_cons,
                           S.fct = Unweighted.Kappa.WkW, S.space.H0 = c(0,1),
                           strata = rep(c(1,2,3,4), each = 9),
                           fixed.strata = "none", trans.g = "[A,B]")

# Constraints (2)
X_cond_RC_v <- c(1,1,0,0,1,0,1,0,1,0,0,0,0,0,1,0,0,0,-1,0,0,
                 1,1,0,0,1,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,
                 1,1,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,
                 1,1,0,0,0,1,1,0,0,1,0,0,0,0,0,0,1,0,-1,0,0,
                 1,1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,
                 1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,
                 1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,-1,0,0,
                 1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,
                 1,0,1,0,1,0,1,0,0,0,1,0,0,0,1,0,0,0,0,-1,0,
                 1,0,1,0,1,0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,
                 1,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,
                 1,0,1,0,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,-1,0,
                 1,0,1,0,0,1,0,1,0,0,0,1,0,0,0,0,0,1,0,0,0,
                 1,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,
                 1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,-1,0,
                 1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,
                 1,0,0,1,1,0,1,0,0,0,0,0,1,0,1,0,0,0,0,0,-1,
                 1,0,0,1,1,0,0,1,0,0,0,0,1,0,0,1,0,0,0,0,0,
                 1,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,
                 1,0,0,1,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0,-1,
                 1,0,0,1,0,1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,
                 1,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,
                 1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,
                 1,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
                 1,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,1,1,1,
                 1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,
                 1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,
                 1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,1,1,1,
                 1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,
                 1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1,
                 1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,
                 1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,
                 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1,-1,-1)
X_cond_RC_mat <- matrix(X_cond_RC_v, ncol = 21, byrow = TRUE)

cond_RC_HLP_fit <- mph.fit(obs.y, L.fct = "logm", L.mean = TRUE,
                           X = X_cond_RC_mat,
                           strata = rep(c(1,2,3,4), each = 9),
                           fixed.strata = "none")
mph.summary(cond_RC_HLP_fit)

library(MASS)
X_cond_RC_U <- Null(X_cond_RC_mat)
cond_RC_MPH_fit <- mph.fit(obs.y, h.fct = function(m) {t(X_cond_RC_U) %*% log(m)},
                           h.mean = TRUE, strata = rep(c(1,2,3,4), each = 9),
                           fixed.strata = "none")
mph.summary(cond_RC_MPH_fit)

BkW_cond_RC <- ci.table(obs.y, h.fct = function(m) {t(X_cond_RC_U) %*% log(m)},
                        h.mean = TRUE, S.fct = Unweighted.Kappa.BkW,
                        S.space.H0 = c(0,1), trans.g = "[A,B]",
                        strata = rep(c(1,2,3,4), each = 9), fixed.strata = "none")
WkW_cond_RC <- ci.table(obs.y, h.fct = function(m) {t(X_cond_RC_U) %*% log(m)},
                        h.mean = TRUE, S.fct = Unweighted.Kappa.WkW,
                        S.space.H0 = c(0,1), trans.g = "[A,B]",
                        strata = rep(c(1,2,3,4), each = 9), fixed.strata = "none")
BkB_cond_RC <- ci.table(obs.y, h.fct = function(m) {t(X_cond_RC_U) %*% log(m)},
                        h.mean = TRUE, S.fct = Unweighted.Kappa.BkB,
                        S.space.H0 = c(0,1), trans.g = "[A,B]",
                        strata = rep(c(1,2,3,4), each = 9), fixed.strata = "none")
WkB_cond_RC <- ci.table(obs.y, h.fct = function(m) {t(X_cond_RC_U) %*% log(m)},
                        h.mean = TRUE, S.fct = Unweighted.Kappa.WkB,
                        S.space.H0 = c(0,1), trans.g = "[A,B]",
                        strata = rep(c(1,2,3,4), each = 9), fixed.strata = "none")



### Construct test-inversion CIs, without additionally imposed constraints.

# V. Binomial success rate parameter p.
#    Model: 0 = x <- X | p ~ Bin(n = 5, p).
#    Goal: Compute approximate 90% CIs for the success probability p.

bin_p_result <- ci.table(c(0, 5), h.fct = 0, S.fct = function(p) {p[1]},
                         S.space.H0 = c(0, 1), cc = 0.9, y.eps = 0.1)
```

```
#    Example 2.1 in Lang (2008).
#    Model: y = (39, 1) <- Y ~ mult(40, p1, p2).
#    Goal: Compute approximate 95% CIs for the success probability p1.

bin_p_eg21_result <- ci.table(c(39,1), h.fct = 0, S.fct = function(p) {p[1]},
                              S.space.H0 = c(0,1), trans.g = "[A,B]")


# VI. Conditional probability.
#     Model: y = (0, 39, 18, 11) <- Y ~ mult(68, p1, p2, p3, p4)
#     Goal: Compute approximate 95% CIs for the conditional probability
#           p1 / (p1 + p2).

cond_prob_result <- ci.table(c(0, 39, 18, 11), h.fct = 0,
                             S.fct = function(p) {p[1] / (p[1] + p[2])},
                             S.space.H0 = c(0, 1), y.eps = 0.1)

#     Model: y = (0, 39 // 18, 11) <- Y ~ prod mult(39, p1, p2 // 29, p3, p4).
#     That is,
#     y <- Y ~ MP(gamma, p | strata = c(1, 1, 2, 2), fixed = "all"),
#          where gamma = (39, 29)'.
#     Goal: Compute approximate 95% CIs for p1.

cond_prob_SS_result <- ci.table(c(0, 39, 18, 11), h.fct = 0,
                                S.fct = function(p) {p[1]}, S.space.H0 = c(0, 1),
                                strata = c(1, 1, 2, 2), y.eps = 0.1)


# VII. Difference between conditional probabilities.
#      Model: y = (0, 39, 18, 11) <- Y ~ mult(68, p1, p2, p3, p4)
#      Goal: Compute approximate 95% CIs for the difference between conditional
#            probabilities, p1 / (p1 + p2) - p3 / (p3 + p4).

diff_cond_prob_result <- ci.table(c(0, 39, 18, 11), h.fct = 0,
                              S.fct = function(p) {p[1]/(p[1]+p[2]) - p[3]/(p[3]+p[4])},
                                  S.space.H0 = c(-1, 1), trans.g = "[A,B]")


# VIII. Gamma variant.
#       Example 2.3 in Lang (2008).
#       Model: y = (25, 25, 12 // 0, 1, 3)
#                   ~ prod mult(62, p11, p12, p13 // 4, p21, p22, p23).
#       Goal: Compute approximate 95% CIs for the Gamma* parameter as
#             described in Lang (2008).

Gamma_variant_23 <- function(p) {
  p <- matrix(p, 2, 3, byrow = TRUE)
  P.case.gt.control <- (p[2, 2] + p[2, 3]) * p[1, 1] + p[2, 3] * p[1, 2]
  P.case.lt.control <- p[1, 2] * p[2, 1] + p[1, 3] * (p[2, 1] + p[2, 2])
  P.case.neq.control <- P.case.gt.control + P.case.lt.control
  P.case.gt.control / P.case.neq.control
}
```

```
Gamma_variant_result <- ci.table(c(25, 25, 12, 0, 1, 3), h.fct = 0,
                                 S.fct = Gamma_variant_23, S.space.H0 = c(0, 1),
                                 trans.g = "[A,B]", strata = c(1, 1, 1, 2, 2, 2))

### Alternative code...
gammastar.fct <- function(p) {
  nr <- nrow(p)
  nc <- ncol(p)
  probC <- 0
  probD <- 0
  for (i in 1:(nr-1)) {
    for (j in 1:(nc-1)) {
      Aij <- 0
      for (h in (i+1):nr) {
        for (k in (j+1):nc) {
          Aij <- Aij + p[h, k]
        }
      }
      probC <- probC + p[i, j] * Aij
    }
  }
  for (i in 1:(nr-1)) {
    for (j in 2:nc) {
      Aij <- 0
      for (h in (i+1):nr) {
        for (k in 1:(j-1)) {
          Aij <- Aij + p[h, k]
        }
      }
      probD <- probD + p[i, j] * Aij
    }
  }
  probC / (probC + probD)
}

Gamma_variant_23_a <- function(p) {
  p <- matrix(p, 2, 3, byrow = TRUE)
  gammastar.fct(p)
}
Gamma_variant_a_result <- ci.table(c(25, 25, 12, 0, 1, 3), h.fct = 0,
                                   S.fct = Gamma_variant_23_a,
                                   S.space.H0 = c(0, 1), trans.g = "[A,B]",
                                   strata = c(1, 1, 1, 2, 2, 2))


# IX. Global odds ratio.
#     Model: y = (25, 25, 12 // 0, 1, 3)
#                  ~ prod mult(62, p11, p12, p13 // 4, p21, p22, p23).
#     Goal: Compute approximate 95% CIs for the first global odds ratio.

global_odds_ratio_23_11 <- function(p) {
  p <- matrix(p, 2, 3, byrow = TRUE)
  p[1, 1] * (p[2, 2] + p[2, 3]) / (p[2, 1] * (p[1, 2] + p[1, 3]))
}
global_odds_ratio_result <- ci.table(c(25, 25, 12, 0, 1, 3), h.fct = 0,
                                     S.fct = global_odds_ratio_23_11,
                                     S.space.H0 = c(0, Inf), trans.g = "log",
                                     strata = c(1, 1, 1, 2, 2, 2))


# X. Difference between product-multinomial probabilities.
#    Example 2.2 in Lang (2008).
#    Source (secondary): Agresti 2002:65
#    Early study of the death penalty in Florida (Radelet)
#    Victim Black...
#    White Defendant  0/9   received Death Penalty
#    Black Defendant  6/103 received Death Penalty
#
#    Model: y = (0, 9 // 6, 97) <- Y ~ prod mult(9, p1, p2 // 103, p3, p4).
#    Goal: Compute approximate 95% CIs for the difference between
#          product-multinomial probabilities, p1 - p3.

diff_prod_mult_prob_result <- ci.table(c(0, 9, 6, 97), h.fct = 0,
                                       S.fct = function(p) {p[1] - p[3]},
                                       S.space.H0 = c(-1, 1),
                                       trans.g = "Fisher's z",
                                       strata = c(1, 1, 2, 2))

### Alternative (artificial) data that is even more sparse...

diff_prod_mult_prob_a_result <- ci.table(c(0, 9, 0, 97), h.fct = 0,
                                         S.fct = function(p) {p[1] - p[3]},
                                         S.space.H0 = c(-1, 1),
                                         trans.g = "Fisher's z",
                                         strata = c(1, 1, 2, 2), y.eps = 0.4)


# XI. Kappa coefficient.
#     Example 2.4 in Lang (2008).
#     Model: y = (4, 0, 0, 0, 1, 0, 0, 0, 15)
#              <- Y ~ mult(20, p11, p12, ..., p33).
#     Goal: Compute approximate 95% CIs for the unweighted kappa coefficient.

Kappa_coeff_33 <- function(p) {
  p <- matrix(p, 3, 3, byrow = TRUE)
  s1 <- p[1, 1] + p[2, 2] + p[3, 3]
  prow <- apply(p, 1, sum)
  pcol <- apply(p, 2, sum)
  s2 <- prow[1] * pcol[1] + prow[2] * pcol[2] + prow[3] * pcol[3]
  (s1 - s2) / (1 - s2)
}
kappa_coeff_result <- ci.table(c(4, 0, 0, 0, 1, 0, 0, 0, 15), h.fct = 0,
                               S.fct = Kappa_coeff_33, S.space.H0 = c(-1, 1))
```

## compute_cons_MLE_ase *Constrained MLE and ASE*

### Description

Computes the constrained MLE of $S_0(m)$ subject to equality constraints $h_0(m) = 0$ under the specified strata and fixed.strata configuration, and its associated asymptotic standard error. Here $m$ is the vector of expected table counts, i.e. $m = E(Y)$.

### Usage

```
compute_cons_MLE_ase(y, strata, fixed.strata, h0.fct, h0.fct.deriv, S0.fct,
                     S0.fct.deriv, max.mph.iter, step, change.step.after,
                     y.eps, iter.orig, norm.diff.conv, norm.score.conv,
                     max.score.diff.iter)
```

### Arguments

| | |
|---|---|
| y | Observed table counts in the contingency table(s), in vector form. |
| strata | Vector of the same length as y that gives the stratum membership identifier. |
| fixed.strata | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| h0.fct | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| h0.fct.deriv | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If h0.fct.deriv is not specified or h0.fct.deriv = NULL, numerical derivatives will be used. |
| S0.fct | The estimand function $S_0(\cdot)$ with respect to $m$. |
| S0.fct.deriv | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If S0.fct.deriv is not specified or S0.fct.deriv = NULL, numerical derivatives will be used. |
| max.mph.iter, step, change.step.after, y.eps, iter.orig, norm.diff.conv, norm.score.conv, max.score.diff.iter | The parameters used in mph.fit. |

### Value

compute_cons_MLE_ase returns a vector of length two. The first element S0.fct.m_H0 is the constrained MLE of $S_0(m)$ subject to equality constraints $h_0(m) = 0$, and the second element ase.S0.fct.m_H0 is the associated asymptotic standard error.

### Author(s)

Qiansheng Zhu

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`ci.table`

---

## create.U — *Orthogonal Complement of the Column Space of a Matrix*

---

### Description

Creates a full column rank matrix, $U$, with column space equal to the orthogonal complement of the column space of $X$. That is, $U$ has column space equal to the null space of $X'$.

### Usage

```
create.U(X)
```

### Arguments

| | |
|---|---|
| `X` | A full column rank matrix. |

### Value

`create.U` returns a full column rank matrix `U`, with column space equal to the orthogonal complement of the column space of `X`.

### Author(s)

Joseph B. Lang

### See Also

`mph.fit`

### Examples

```
X <- matrix(seq(1, 12), ncol = 2, byrow = TRUE)
create.U(X)
```

## create.Z.ZF *Population Matrix and Sampling Constraint Matrix*

### Description

Creates the population (aka strata) matrix $Z$ and the sampling constraint matrix $Z_F$ using strata and sampling constraint information found in input variables `strata` and `fixed.strata`.

### Usage

```
create.Z.ZF(strata, nrowZ = length(strata), fixed.strata = "all")
```

### Arguments

| | |
|---|---|
| `strata` | The vector that gives the stratum membership identifier. |
| `nrowZ` | Number of rows of the to-be-created population (aka strata) matrix $Z$. |
| `fixed.strata` | The object that gives information on which stratum have fixed sample sizes. It can equal one of the keywords, `fixed.strata = "all"` or `fixed.strata = "none"`, or it can be a vector of stratum membership identifiers, e.g. `fixed.strata = c(1,3)`. Default: `fixed.strata = "all"`. |

### Value

`create.Z.ZF` returns a list, which includes the following two objects:

| | |
|---|---|
| `Z` | Population (aka strata) matrix. |
| `ZF` | Sampling constraint matrix. |

### Author(s)

Joseph B. Lang

### References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

### See Also

`mph.fit`

### Examples

```
create.Z.ZF(c(1, 1, 2, 3, 3), fixed.strata = "all")
create.Z.ZF(c(1, 1, 2, 3, 3), fixed.strata = "none")
create.Z.ZF(c(1, 1, 2, 3, 3), fixed.strata = c(1, 2))
```

## diff_Gsq_nr *Difference in G-Squared Statistic Based CIs (Non-Robust)*

### Description

Constructs confidence intervals (CIs), based on the difference in $G^2$ statistic, for estimands in contingency tables subject to equality constraints.

These confidence intervals are also referred to as likelihood ratio confidence intervals or profile likelihood confidence intervals.

The program may stop because of a non-convergence issue.

### Usage

```
diff_Gsq_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv, S0.fct,
            S0.fct.deriv, max.mph.iter, step, change.step.after,
            y.eps, iter.orig, norm.diff.conv, norm.score.conv,
            max.score.diff.iter, S.space.H0, tol.psi, tol,
            max.iter, cut.off, delta)
```

### Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |

| | |
|---|---|
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |

## Value

Provided that `diff_Gsq_nr` does not stop, it returns a 1-by-2 matrix which displays two endpoints of the confidence interval based on the difference in $G^2$ statistic.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`diff_Gsq_robust`, `f.psi`, `ci.table`

---

`diff_Gsq_robust` *Difference in G-Squared Statistic Based CIs (Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the difference in $G^2$ statistic, for estimands in contingency tables subject to equality constraints.

These confidence intervals are also referred to as likelihood ratio confidence intervals or profile likelihood confidence intervals.

## Usage

```
diff_Gsq_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                S0.fct, S0.fct.deriv, max.mph.iter, step,
                change.step.after, y.eps, iter.orig, norm.diff.conv,
                norm.score.conv, max.score.diff.iter, S.space.H0,
                tol.psi, tol, max.iter, cut.off, delta, adj.epsilon,
                iter.robust.max, iter.robust.eff)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |

| | |
|---|---|
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `adj.epsilon`, `iter.robust.max`, `iter.robust.eff` | The parameters used in the robustifying procedure. |

## Value

`diff_Gsq_robust` returns a list, which includes two objects. The first object is a 1-by-2 matrix which displays two endpoints of the confidence interval based on the difference in $G^2$ statistic. For the second object, it includes the warning message that occurs during construction of the confidence interval if the robustifying procedure is evoked: `"diff.Gsq.CI: Adjustment used. Not on original data.\n"`. If the robustifying procedure is not evoked, the second object is `NULL`.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`diff_Gsq_nr`, `f.psi`, `ci.table`

`diff_PD_nr` *Difference in Power-Divergence Statistic Based CIs (Non-Robust)*

## Description

Constructs confidence intervals (CIs), based on the difference in power-divergence statistic, for estimands in contingency tables subject to equality constraints.

The program may stop because of a non-convergence issue.

## Usage

```
diff_PD_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv, S0.fct,
           S0.fct.deriv, max.mph.iter, step, change.step.after,
           y.eps, iter.orig, norm.diff.conv, norm.score.conv,
           max.score.diff.iter, S.space.H0, tol.psi, tol,
           max.iter, cut.off, delta, pdlambda)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. |

## Value

Provided that `diff_PD_nr` does not stop, it returns a 1-by-2 matrix which displays two endpoints of the confidence interval based on the difference in power-divergence statistic.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`diff_PD_robust`, `f.psi`, `ci.table`

---

`diff_PD_robust` *Difference in Power-Divergence Statistic Based CIs (Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the difference in power-divergence statistic, for estimands in contingency tables subject to equality constraints.

## Usage

```
diff_PD_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
               S0.fct, S0.fct.deriv, max.mph.iter, step,
               change.step.after, y.eps, iter.orig, norm.diff.conv,
               norm.score.conv, max.score.diff.iter, S.space.H0,
               tol.psi, tol, max.iter, cut.off, delta, pdlambda,
               adj.epsilon, iter.robust.max, iter.robust.eff)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |

| | |
|---|---|
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m.$ If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. |
| `adj.epsilon`, `iter.robust.max`, `iter.robust.eff` | The parameters used in the robustifying procedure. |

## Value

`diff_PD_robust` returns a list, which includes two objects. The first object is a 1-by-2 matrix which displays two endpoints of the confidence interval based on the difference in power-divergence statistic. For the second object, it includes the warning message that occurs during construction of the confidence interval if the robustifying procedure is evoked: `"diff.PD.CI: Adjustment used. Not on original data.\n"`. If the robustifying procedure is not evoked, the second object is `NULL`.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`diff_PD_nr`, `f.psi`, `ci.table`

## diff_Xsq_nr *Difference in $X$-Squared Statistic Based CIs (Non-Robust)*

### Description

Constructs confidence intervals (CIs), based on the difference in $X^2$ statistic, for estimands in contingency tables subject to equality constraints.

The program may stop because of a non-convergence issue.

### Usage

```
diff_Xsq_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
            S0.fct, S0.fct.deriv, max.mph.iter, step,
            change.step.after, y.eps, iter.orig, norm.diff.conv,
            norm.score.conv, max.score.diff.iter, S.space.H0,
            tol.psi, tol, max.iter, cut.off, delta)
```

### Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |

## Value

Provided that `diff_Xsq_nr` does not stop, it returns a 1-by-2 matrix which displays two endpoints of the confidence interval based on the difference in $X^2$ statistic.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`diff_Xsq_robust`, `f.psi`, `ci.table`

---

`diff_Xsq_robust` *Difference in $X$-Squared Statistic Based CIs (Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the difference in $X^2$ statistic, for estimands in contingency tables subject to equality constraints.

## Usage

```
diff_Xsq_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                S0.fct, S0.fct.deriv, max.mph.iter, step,
                change.step.after, y.eps, iter.orig, norm.diff.conv,
                norm.score.conv, max.score.diff.iter, S.space.H0,
                tol.psi, tol, max.iter, cut.off, delta, adj.epsilon,
                iter.robust.max, iter.robust.eff)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |

S0.fct.deriv — The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m.$ If S0.fct.deriv is not specified or S0.fct.deriv = NULL, numerical derivatives will be used.

max.mph.iter, step, change.step.after, y.eps, iter.orig, norm.diff.conv, norm.score.conv, max.score.diff.iter — The parameters used in mph.fit.

S.space.H0 — Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints.

tol.psi, tol, max.iter — The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation.

cut.off — qchisq(cc, 1). i.e. The chi-square cutoff, with 1 df, based on the significance level 1-cc.

delta — The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step.

adj.epsilon, iter.robust.max, iter.robust.eff — The parameters used in the robustifying procedure.

## Value

diff_Xsq_robust returns a list, which includes two objects. The first object is a 1-by-2 matrix which displays two endpoints of the confidence interval based on the difference in $X^2$ statistic. For the second object, it includes the warning message that occurs during construction of the confidence interval if the robustifying procedure is evoked: "diff.Xsq.CI: Adjustment used. Not on original data.\n". If the robustifying procedure is not evoked, the second object is NULL.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

diff_Xsq_nr, f.psi, ci.table

f.psi *Model Comparison Statistics*

## Description

Computes one of the model comparison statistics.

The model comparison statistics include:

- "`diff.Gsq`": The difference in $G^2$ statistic,
$$G^2(\psi) - G^2 = G^2(y; H_0(\psi)) - G^2(y; H_0);$$
- "`diff.Xsq`": The difference in $X^2$ statistic,
$$X^2(\psi) - X^2 = X^2(y; H_0(\psi)) - X^2(y; H_0);$$
- "`diff.PD`": The difference in power-divergence statistic, with index parameter $\lambda$,
$$PD_\lambda(\psi) - PD_\lambda = PD_\lambda(y; H_0(\psi)) - PD_\lambda(y; H_0);$$
- "`nested.Gsq`": The nested $G^2$ statistic,
$$G^2(y; H_0(\psi)|H_0);$$
- "`nested.Xsq`": The nested $X^2$ statistic,
$$X^2(y; H_0(\psi)|H_0);$$
- "`nested.PD`": The nested power-divergence statistic, with index parameter $\lambda$,
$$PD_\lambda(y; H_0(\psi)|H_0).$$

## Usage

```
f.psi(y, strata, fixed.strata, h0.fct, h0.fct.deriv = NULL,
      S0.fct, S0.fct.deriv = NULL, method_specific, psi,
      max.mph.iter, step, change.step.after, y.eps, iter.orig,
      norm.diff.conv, norm.score.conv, max.score.diff.iter,
      pdlambda = NULL, Gsq_H0, Xsq_H0, PD_H0, cons.MLE.m_H0)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |

| | |
|---|---|
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `method_specific` | A character string that indicates which model comparison statistic to compute. It can be one of `"diff.Xsq"`, `"nested.Xsq"`, `"diff.Gsq"`, `"nested.Gsq"`, `"diff.PD"`, or `"nested.PD"`. |
| `psi` | The real number $\psi$ in the model comparison statistic. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. |
| `Gsq_H0` | The $G^2$ statistic for testing $H_0$ vs. not $H_0$, i.e. $G^2(y; H_0)$. |
| `Xsq_H0` | The $X^2$ statistic for testing $H_0$ vs. not $H_0$, i.e. $X^2(y; H_0)$. |
| `PD_H0` | The power-divergence statistic for testing $H_0$ vs. not $H_0$, i.e. $PD_\lambda(y; H_0)$. |
| `cons.MLE.m_H0` | Constrained MLE of $m = E(Y)$ subject to $H_0$. |

## Value

`f.psi` returns a numeric value, which is the computed model comparison statistic.

## Note

Among the four inputs: `Gsq_H0`, `Xsq_H0`, `PD_H0`, and `cons.MLE.m_H0`, only one of them needs to be specified.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`diff_Xsq_nr`, `nested_Xsq_nr`, `diff_Gsq_nr`, `nested_Gsq_nr`, `diff_PD_nr`, `nested_PD_nr`, `diff_Xsq_robust`, `nested_Xsq_robust`, `diff_Gsq_robust`, `nested_Gsq_robust`, `diff_PD_robust`, `nested_PD_robust`, `ci.table`

## M.fct — *Marginalizing Matrix Based on Strata Information*

### Description

Creates the marginalizing matrix $M$ using strata information found in input variable `strata`. That is, $Mp$ gives the marginal probabilities corresponding to the levels of factor `strata`.

### Usage

```
M.fct(strata, ncells = length(strata))
```

### Arguments

| | |
|---|---|
| `strata` | The vector that gives the stratum membership identifier. |
| `ncells` | Number of contingency table cells. Default: `ncells = length(strata)`. |

### Value

`M.fct` returns the marginalizing matrix `M`.

### Note

Marginals are ordered according to the levels of factor `strata`.

Examples:

```
V1   V2   y
 b  yes  15
 a   no  12
 a  yes  13
 b  yes   5
 b   no   1
```

```
M1 <- M.fct(V1)
M1 %*% y
```

```
     [,1]
a      25
b      21
```

```
M2 <- M.fct(V2)
M2 %*% y
```

```
      [,1]
no      13
yes     33
```

```
M12 <- M.fct(paste(V1, V2))
M12 %*% y
```

```
         [,1]
a   no     12
a  yes     13
b   no      1
b  yes     20
```

### Author(s)

Joseph B. Lang

### References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

### Examples

```
M.fct(rep(1, 9))
M.fct(seq(1, 9))
M.fct(c(1, 1, 2, 3, 3))
```

---

## mph.fit *Fitting MPH and HLP Models*

---

### Description

Computes maximum likelihood estimates and fit statistics for multinomial-Poisson homogeneous (MPH) and homogeneous linear predictor (HLP) models for contingency tables.

More detailed DOCUMENTATION and EXAMPLES of `mph.fit` are online.

### Usage

```
mph.fit(y, h.fct = constraint, constraint = NULL, h.mean = FALSE,
        L.fct = link, link = NULL, L.mean = FALSE, X = NULL,
        strata = rep(1, length(y)), fixed.strata = "all",
        check.homog.tol = 1e-9, check.HLP.tol = 1e-9, maxiter = 100,
        step = 1, change.step.after = 0.25 * maxiter, y.eps = 0,
        iter.orig = 5, m.initial = y, norm.diff.conv = 1e-6,
        norm.score.conv = 1e-6, max.score.diff.iter = 10,
        derht.fct = NULL, derLt.fct = NULL, pdlambda = 2/3,
        verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `y` | Vector (not matrix) of table counts. |
| `h.fct` | Function object that defines the constraint function $h(\cdot)$. It must return a column vector. `h.fct` can also be set to `0`, in which case $h(\cdot)$ is viewed as the 0 function, so no constraints are imposed.<br>By default, $h(\cdot)$ is viewed as a function of the table probabilities $p$. If `h.mean` is set to `h.mean = TRUE`, then $h(\cdot)$ is viewed as a function of the expected counts $m$.<br>Default: `h.fct = NULL`. If both `h.fct = NULL` and `L.fct = NULL`, then `h.fct` is set to `0` and no constraints are imposed. If `h.fct = NULL` and `L.fct` is not `NULL`, then `h.fct` will be computed as `t(U) %*% L.fct`. |
| `constraint` | Alias for the argument `h.fct`. Argument `constraint` is secondary to the primary argument `h.fct` in the following senses: If `constraint` and `h.fct` are not equal, `h.fct` is used. |
| `h.mean` | Logical argument. If `h.mean = FALSE` (the default), `h.fct` is treated as a function of $p$. If `h.mean = TRUE`, then `h.fct` is treated as a function of $m$. |
| `L.fct` | Function object that defines the link $L(\cdot)$ in the HLP model; it must return a column vector. Or ... `L.fct = keyword`, where candidate keywords include `"logp"` and `"logm"`.<br>Entering `L.fct = "logp"` tells the program to create the function object as `L.fct <- function(p) {log(p)}`. `L.fct = "logm"` tells the program to (i) create the function object as `L.fct <- function(m) {log(m)}` and (ii) set `L.mean = TRUE`.<br>By default, `L.fct` is treated as a function of the table probabilities $p$ (even if the argument in the `L.fct` function object is `m` ). If `L.mean` is set to `L.mean = TRUE`, then `L.fct` is treated as a function of the expected counts $m$. Default: `L.fct = NULL` means no constraints of the form $L(p) = X\beta$ are imposed. |
| `link` | Alias for the argument `L.fct`. Argument `link` is secondary to the primary argument `L.fct` in the following senses: If `link` and `L.fct` are not equal, `L.fct` is used. |
| `L.mean` | Logical argument. If `L.mean = FALSE` (the default), `L.fct` is treated as a function of $p$. If `L.mean = TRUE`, `L.fct` is treated as a function of $m$. |
| `X` | HLP model design matrix, assumed to be full rank. Default: `X = NULL`; i.e., it is left unspecified and unused. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. Default: `strata = rep(1, length(y))`; i.e. the default is the single stratum (non-stratified) setting. Examples: `strata = A`, or `strata = c(1,1,1,2,2,2,3,3,3)`, or `strata = paste(sep = "", "A=", A, ", B=", B)`. |
| `fixed.strata` | The object that gives information on which stratum have fixed sample sizes. It can equal one of the keywords, `fixed.strata = "all"` or `fixed.strata = "none"`, or it can be a vector of stratum membership identifiers, e.g. `fixed.strata` |

= `c(1,3)` or `fixed.strata = c("pop1", "pop5")`. Default: `fixed.strata = "all"`.

| | |
|---|---|
| `check.homog.tol` | Round-off tolerance for $Z$ homogeneity check. Default: `check.homog.tol = 1e-9`. |
| `check.HLP.tol` | Round-off tolerance for HLP link status check. Default: `check.HLP.tol = 1e-9`. |
| `maxiter` | Maximum number of iterations. Default: `maxiter = 100`. |
| `step` | Step-size value. Default: `step = 1`. |
| `change.step.after` | If the score value increases for more than `change.step.after` iterations in a row, then the initial step size is halved. Default: `change.step.after = 0.25 * maxiter`. |
| `y.eps` | Non-negative constant to be temporarily added to the original counts in `y`. Default: `y.eps = 0`. |
| `iter.orig` | Iteration at which the original counts will be used. Default: `iter.orig = 5`. |
| `m.initial` | Initial estimate of $m$. Default: `m.initial = y`. See Input Note 6 below. |
| `norm.diff.conv` | Convergence criteria value; see `norm.diff` in the **Value** section. Default: `norm.diff.conv = 1e-6`. |
| `norm.score.conv` | Convergence criteria value; see `norm.score` in the **Value** section. Default: `norm.score.conv = 1e-6`. |
| `max.score.diff.iter` | The variable `score.diff.iter` keeps track of how long `norm.score` is smaller than `norm.score.conv`, but `norm.diff` is greater than `norm.diff.conv`. If this is the case too long (i.e. `score.diff.iter >= max.score.diff.iter`), then stop the iterations because the solution likely includes at least one ML fitted value of 0. Default: `max.score.diff.iter = 10`. |
| `derht.fct` | Function object that computes analytic derivative of the transpose of the constraint function $h(\cdot)$ with respect to $m$. If $h(\cdot)$ maps from $R^p$ to $R^q$ (i.e. there are $q$ constraints), then `derht.fct` returns a $p$-by-$q$ matrix of partial derivatives. Default: `derht.fct = NULL`. This means that the derivative is calculated numerically. |
| `derLt.fct` | Function object that computes analytic derivative of the transpose of the link function $L(\cdot)$ with respect to $m$. If $L(\cdot)$ maps from $R^p$ to $R^q$ (i.e. there are $q$ link components), then `derLt.fct` returns a $p$-by-$q$ matrix of partial derivatives. Default: `derLt.fct = NULL`, i.e. by default this derivative is calculated numerically. |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. |
| `verbose` | Logical argument. If `verbose = FALSE`, do not print out iteration information. If `verbose = TRUE`, then iteration information is printed out. Default: `verbose = FALSE`. |

## Details

In the following, let $y$ be the vector of contingency table counts, $p$ be the unknown vector of contingency table probabilities, $s$ be a vector of strata identifiers, and $F$ be the set of strata with a priori fixed sample sizes.

Although `mph.fit` can fit more general models (see below), two important special cases include:

- MPH (Special-Case): $y$ is a realization of random vector $Y$, where $Y \sim MP(\gamma, p|\texttt{strata} = s, \texttt{fixed} = F)$, $h(p) = 0$.
- HLP (Special-Case): $y$ is a realization of random vector $Y$, where $Y \sim MP(\gamma, p|\texttt{strata} = s, \texttt{fixed} = F)$, $L(p) = X\beta$.

Here, $h(\cdot)$ is a smooth constraint function and $L(\cdot)$ is a smooth link function. It is assumed that the constraints in $h(p) = 0$ are non-redundant so that the Jacobian, $\partial h'(p)/\partial p$, is full column rank.

The link $L(\cdot)$ is allowed to be many-to-one and row-rank deficient, so this HLP model is quite general. It is only required that the implied constraints, $U'L(p) = 0$, where $null(U') = span(X)$, are non-redundant.

Here, MP stands for the multinomial-Poisson distribution. The parameters are $\gamma$, the vector of expected sample sizes, and $p$, the vector of table probabilities.

The notation

$$Y \sim MP(\gamma, p|\texttt{strata} = s, \texttt{fixed} = F)$$

means that the random vector $Y$ is composed of independent blocks of multinomial and/or Poisson random variables. The strata vector $s$ determines the blocks and $F$ determines which blocks are multinomial and which blocks comprise independent Poisson random variables. More specifically, suppose there are $K$ strata, so $s$ contains $K$ distinct strata identifiers. The components in $Y$ corresponding to $s = $ `identifier[k]` make up a block. If `identifier[k]` is included in $F$, then this block has a multinomial distribution and $\gamma_k$ is the a priori known, i.e. fixed, sample size. If `identifier[k]` is not in $F$, then this block comprises independent Poisson random variables and $\gamma_k$ is an unknown expected sample size.

Note: Given the observed counts $y$, the pair `(strata, fixed)` $= (s, F)$ contains the same information as the sampling plan triple $(Z, Z_F, n_F)$ described in Lang (2004, 2005). Specifically, $Z = Z(s)$, the strata/population matrix, is determined by $s$. $Z_F = Z_F(s, F)$, the sampling constraint matrix, is determined by $s$ and $F$. $n_F = Z_F'y$ is the vector of a priori fixed sample sizes.

Special case MP distributions include...

- Full Multinomial: $MP(\gamma, p|$`strata = 1, fixed = "all"`$)$. A simple random sample of fixed size $\gamma$ is taken from a single strata (population).
- Product Multinomial: $MP(\gamma, p|$`strata = s, fixed = "all"`$)$. A stratified random sample of fixed sample sizes $\gamma = (\gamma_1, \ldots, \gamma_K)'$ is taken from the $K$ strata determined by $s$.
- Full Poisson: $MP(\gamma, p|$`strata = 1, fixed = "none"`$)$. A simple random sample is taken from a single strata (population). The sample size is random and follows a Poisson distribution with unknown mean $\gamma$.
- Product Poisson: $MP(\gamma, p|$`strata = s, fixed = "none"`$)$. A stratified random sample is taken from $K$ strata. The sample sizes are all random and distributed as Poissons with unknown means in $\gamma = (\gamma_1, \ldots, \gamma_K)'$.

Specifying the MP distribution in `mph.fit`...

The user need only enter `(strata, fixed.strata)`, the input variables corresponding to $(s, F)$. Keywords, `fixed.strata = "all"` [`"none"`] means that all [none] of the strata have a priori fixed sample sizes.

To fit MPH (Special Case), the user must enter the counts `y`, the constraint function `h.fct` (alias `constraint`), and the sampling plan variables, `strata` and `fixed.strata`. Note: The user can omit the sampling plan variables if the default, multinomial sampling `(strata = 1, fixed = "all")`, can be assumed.

To fit HLP (Special Case), the user must enter the counts `y`, the link function `L.fct` (alias `link`), the model matrix `X`, and the sampling plan variables, `strata` and `fixed.strata`. Note: The user can omit the sampling plan variables if the default, multinomial sampling, can be assumed.

IMPORTANT: When specifying the model and creating the input objects for `mph.fit`, keep in mind that the interpretation of the table probabilities $p$ depends on the sampling plan!

Specifically, if the $i^{th}$ count $y_i$ is in block $k$ (i.e. corresponds with strata `identifier[k]`), then the $i^{th}$ table probability $p_i$ is the conditional probability defined as $p_i$ = probability of a Type $i$ outcome GIVEN that the outcome is one of the types in stratum $k$.

For example, in an $I$-by-$J$ table with row variable $A$ and column variable $B$, if row-stratified sampling is used, the table probabilities have the interpretation, $p_{ij}$ = prob of a Type $(i, j)$ outcome GIVEN that the outcome is one of the types in stratum $i$ (i.e. one of $(i, 1), \ldots, (i, J)) = P(A = i, B = j|A = i) = P(B = j|A = i)$. For column-stratified sampling, $p_{ij} = P(A = i|B = j)$. And for non-stratified sampling, $p_{ij} = P(A = i, B = j)$.

Log-Linear Models: Log-linear models specified as $\log(p) = X\beta$, are HLP models.

As with any HLP model, $\log(p) = X\beta$ can be restated as a collection of constraints; specifically, $\log(p) = X\beta$ is equivalent to $h(p) = U' \log(p) = 0$, where $null(U') = span(X)$. Noting that $Z'p = 1$, we see that to avoid redundant constraints, $span(X)$ should contain $span(Z)$. Loosely, fixed-by-sampling-design parameters should be included.

Log-linear models of the form $\log(p) = X\beta$ are simple to fit using `mph.fit`. For example,
`> mph.fit(y, link = "logp", X = model.matrix(~ A + B))`,
or, equivalently,
`> mph.fit(y, link = function(p) {log(p)}, X = model.matrix(~ A + B))`.

MORE GENERAL MPH and HLP MODELS...

Instead of $(\gamma, p)$, the MP distribution can alternatively be parameterized in terms of the vector of expected table counts, $m = E(Y)$. Formally, $(\gamma, p)$ and $m$ are in one-to-one correspondence and satisfy:

$$m = Diag(Z\gamma)p,$$

and

$$\gamma = Z'm, p = Diag^{-1}(ZZ'm)m.$$

Here, $Z = Z(s)$ is the $c$-by-$K$ strata/population matrix determined by strata vector $s$. Specifically, $Z_{ik} = I\{s_i = \texttt{identifier[k]}\}$.

The MPH (Special-Case) Model given above is a special case because it constrains the expected counts $m$ only through the table probabilities $p$. Similarly, the HLP (Special-Case) Model given above is a special case because it uses a link function that depends on $m$ only through the table probabilities $p$.

More generally, `mph.fit` computes maximum likelihood estimates and fit statistics for MPH and HLP models of the form...

- MPH: $y$ is a realization of random vector $Y$, where $Y \sim MP(\gamma, p|\texttt{strata} = s, \texttt{fixed} = F), h(m) = 0$.
- HLP: $y$ is a realization of random vector $Y$, where $Y \sim MP(\gamma, p|\texttt{strata} = s, \texttt{fixed} = F), L(m) = X\beta$.

Here, $h(\cdot)$ is a smooth constraint function that must also be $Z$ (i.e. strata $s$) homogeneous. $L(\cdot)$ is a smooth link function that must also satisfy the HLP conditions with respect to $Z$ (i.e. strata $s$) and $X$. That is,

- (1) $L(\cdot)$ has HLP link status with respect to $Z$, and
- (2) The implied constraint function $h(m) = U'L(m)$ is $Z$ homogeneous. Here, $null(U') = span(X)$.

Here, (1) $L(\cdot)$ has HLP link status with respect to $Z$ if, for $m = Diag(Z\gamma)p$,

- (1)(a) $L(m) = a(\gamma) + L(p)$, where $a(\gamma_1/\gamma_2) - a(1) = a(\gamma_1) - a(\gamma_2)$, i.e. $a(\gamma)$ has the form $C \log \gamma + \texttt{constant}$; or
- (1)(b) $L(m) = G(\gamma)L(p)$, where $G(\gamma)$ is a diagonal matrix with diagonal elements that are powers of the $\gamma$ elements, i.e. $L(\cdot)$ is $Z$ homogeneous (see Lang (2004)); or
- (1)(c) The components of $L(\cdot)$ are a mixture of types (a) and (b): $L_j(m) = a_j(\gamma) + L_j(p)$ or $L_j(m) = G_j(\gamma)L_j(p), j = 1, \ldots, l$.

Lang (2005) described HLP models that satisfied (1)(a) and (2), but the definition of HLP models can be broadened to include those models satisfying (1) and (2). That is, HLP models can be defined so they also include models that satisfy (1)(b) and (2) or (1)(c) and (2). `mph.fit` uses this broader definition of HLP Model.

Note: The input variable `h.mean` must be set to `TRUE` to fit this more general MPH model. Similarly, the input variable `L.mean` must be set to `TRUE` to fit this more general HLP model. (An exception: If the link function is specified using the keyword `"logm"` then `L.mean` is automatically set to `TRUE`.)

Note: `mph.fit` carries out "necessary-condition" checks of $Z$ homogeneity of $h(\cdot)$ and HLP link status of $L(\cdot)$ for these general models.

Log-Linear Models: Log-linear models of the form $\log(m) = X\beta$ are HLP models provided the $span(X)$ contains the $span(Z)$. Loosely, provided fixed-by-design parameters are included, the log-linear model is a special case HLP model.

Log-linear models of the form $\log(m) = X\beta$ are simple to fit using `mph.fit`. For example,
`> mph.fit(y, link = "logm", X = model.matrix(~ A + B))`,
or, equivalently,
`> mph.fit(y, link = function(m) {log(m)}, L.mean = TRUE, X = model.matrix(~ A + B))`.

Note: Most reasonable generalized log-linear models, which have the form $L(m) = C \log Mm = X\beta$, are also HLP models. See Lang (2005).

## Value

`mph.fit` returns a list, which includes the following objects:

| | |
|---|---|
| `y` | Vector of counts used in the algorithm for ML estimation. Usually, this vector is identical to the observed table counts. |
| `m` | Vector of ML fitted values. |
| `covm` | Approximate covariance matrix of fitted values. |
| `p` | Vector of cell probability ML estimates. |
| `covp` | Approximate covariance matrix of cell probability estimators. |
| `lambda` | Vector of Lagrange multiplier ML estimates. |
| `covlambda` | Approximate covariance matrix of multiplier estimators. |
| `resid` | Vector of raw residuals (i.e. observed minus fitted counts). |
| `presid` | Vector of Pearson residuals. |
| `adjresid` | Vector of adjusted residuals. |
| `covresid` | Approximate covariance matrix of raw residuals. |
| `Gsq` | Likelihood ratio statistic for testing $H_0 : h(m) = 0$ vs. $H_1$ : not $H_0$. |
| `Xsq` | Pearson's score statistic (same as Lagrange multiplier statistic) for testing $H_0 : h(m) = 0$ vs. $H_1$ : not $H_0$. |
| `Wsq` | Generalized Wald statistic for testing $H_0 : h(m) = 0$ vs. $H_1$ : not $H_0$. |
| `PD.stat` | Power-divergence statistic (with index parameter `pdlambda`) for testing $H_0 : h(m) = 0$ vs. $H_1$ : not $H_0$. |
| `df` | Degrees of freedom associated with `Gsq`, `Xsq`, and `PD.stat`. $df = \dim(h)$. |
| `beta` | Vector of HLP model parameter ML estimates. |
| `covbeta` | Approximate covariance matrix of HLP model parameter estimators. |
| `L` | Vector of HLP model link ML estimates. |
| `Lobs` | Vector of HLP model observed link values, $L(y)$. |
| `covL` | Approximate covariance matrix of HLP model link estimators. |
| `Lresid` | Vector of adjusted link residuals of the form $$(L(\texttt{obs}) - L(\texttt{fitted}))/ase(L(\texttt{obs}) - L(\texttt{fitted})).$$ |
| `iter` | Number of update iterations performed. |
| `norm.diff` | Distance between last and second last `theta` iterates (`theta` is the vector of log fitted values and Lagrange multipliers). |
| `norm.score` | Distance between the score vector and zero. |
| `h.fct` | Constraint function used in algorithm. |
| `h.input.fct` | Constraint function as originally input. |
| `h.mean` | Logical variable. If `h.mean = FALSE`, `h.fct` is treated as a function of $p$. If `h.mean = TRUE`, `h.fct` is treated as a function of $m$. |
| `derht.fct` | Analytic function used in algorithm that computes derivative of transpose of constraint function $h$. |
| `L.fct` | Link function used in algorithm. |
| `L.input.fct` | Link function as originally input. |

| | |
|---|---|
| `L.mean` | Logical variable. If `L.mean = FALSE`, `L.fct` is treated as a function of $p$. If `L.mean = TRUE`, `L.fct` is treated as a function of $m$. |
| `derLt.fct` | Analytic function used in algorithm that computes derivative of transpose of link function $L$. |
| `X` | HLP model design matrix used in algorithm. |
| `U` | Orthogonal complement of design matrix $X$. |
| `triple` | A list object containing the sampling plan triple $(Z, Z_F, n)$, where $Z$ is the population (or strata) matrix, $Z_F$ is the sampling constraint matrix, and $n$ is the collection of fixed sample sizes. |
| `strata` | `strata` variable used as input. |
| `fixed.strata` | The strata corresponding to fixed sample sizes. |
| `warn.message` | Message stating whether or not the original counts were used. |
| `warn.message.score` | Message stating whether or not the norm score convergence criterion was met. |
| `warn.message.diff` | Message stating whether or not the norm diff convergence criterion was met. |

## Note

Input Notes:

1. CONSTRAINT FUNCTION.

   `constraint` is an alias for `h.fct`. If `h.fct` is a function object, it must return a column vector.

   By default, `h.fct` is treated as a function of the table probabilities $p$. To treat `h.fct` as a function of the expected counts $m$, you must set `h.mean = TRUE` (by default, `h.mean = FALSE`).

   The fitting algorithm will fail if the constraints in `h.fct` $= 0$ are redundant.

2. MODEL WITH NO CONSTRAINTS.

   The model with no constraints can be specified using `h.fct = 0`. The no-constraint model is the default when neither `h.fct` nor `L.fct` are input (i.e. when `h.fct = NULL` and `L.fct = NULL`).

3. HLP MODEL SPECIFICATION.

   `link` is an alias for `L.fct`. For HLP models, both `L.fct` and `X` must be specified. The design matrix $X$ must be of full column rank. `mph.fit` recognizes two keywords for link specification, `"logp"` and `"logm"`. These are convenient for log-linear modeling. If `L.fct` is a function object, it must return a column vector.

   By default, `L.fct` is treated as a function of the table probabilities $p$. To treat `L.fct` as a function of the expected counts $m$, you must set `L.mean = TRUE` (by default, `L.mean = FALSE`).

   The constraint function `h.fct` is typically left unspecified for HLP models, but it need not be.

   If `h.fct` is left unspecified, it is created within the program as `h.fct(m) <- function(m) {t(U) %*% L.fct(m)}`, where matrix $U$ is an orthogonal complement of $X$. If `h.fct` is specified, the constraints implied by `L.fct`$(p) = X\beta$, or `L.fct`$(m) = X\beta$, are ignored.

   Note: Although the HLP constraints are ignored when `h.fct` is specified, estimates of $\beta$ and the link are computed under the model with constraints `h.fct`$(p) = 0$ or `h.fct`$(m) = 0$.

   The fitting algorithm will fail to converge if the implied constraints, $U'$ `L.fct` $= 0$, include redundancies.

4. EXTENDED ML ESTIMATES.

   When ML estimates are non-existent owing to zero counts, `norm.diff` will not converge to zero, instead it tends to level off at some constant positive value. This is because at least one ML fitted value is $0$, which on the log scale is $\log(0) = -\infty$, and the log-scale iterates slowly move toward $-\infty$. One solution to this problem is to set the convergence value `norm.diff.conv` to some large number so only the score convergence criterion is used. In this case, the algorithm often converges to a solution that can be viewed as an extended ML estimate, for which $0$ estimates are allowed. `mph.fit` automates the detection of such problems. See the description of the input variable `max.score.diff.iter` above and the MISC COMPUTATIONAL NOTES in `mph.fit` online documentation.

5. CONVERGENCE PROBLEMS / FINE TUNING ITERATIONS.

   First check to make sure that the model is correctly specified and redundant constraints are avoided.

   When ML estimates exist, but there are non-convergence problems (perhaps caused by zero counts), a modification to the tuning parameters `step`, `change.step.after`, `y.eps`, and/or `iter.orig` will often lead to convergence.

   With zero counts, it might help to set `y.eps = 0.1` (or some other positive number) and `iter.orig = 5` (the default). This tells the program to initially use `y + y.eps` rather than the original counts `y`. At iteration $5 =$ `iter.orig`, after the algorithm has had time to move toward a non-boundary solution, the original counts are again used.

   To further mitigate non-convergence problems, the parameter `step` can be set to a smaller value (default: `step = 1`) so the iterates do not change as much.

6. The initial estimate of $m$ is actually `m.initial + y.eps + 0.01 * ((m.initial + y.eps) == 0)`. The program defaults are `m.initial = y` and `y.eps = 0`. Note: If `m.initial > 0` and `y.eps = 0`, then the initial estimate of $m$ is simply `m.initial`.

Output Notes:

1. ITERATION HISTORY.

   An iteration history is printed out when `verbose` is set equal to `TRUE`. A single line of the history looks like the following:

   `"iter= 18[0] norm.diff= 3.574936e-08 norm.score= 1.901705e-15"`.

   Here, `iter` is the number of update iterations performed. The number in `[]` gives the number of step size searches required within each iteration. `norm.diff` and `norm.score` are defined above. Finally, the time elapsed is output. Note: For the model with no restrictions (`h.fct = 0`), there are no step size changes.

2. STORING and VIEWING RESULTS.

   To store the results of `mph.fit`, issue a command like the following example

   `> results <- mph.fit(y, h.fct = h.fct)`

   Use program `mph.summary` to view the results of `mph.fit`. Specifically, if the results of `mph.fit` are saved in object `results`, submit the command `mph.summary(results)` [or `mph.summary(results, TRUE)` or `mph.summary(results, TRUE, TRUE)` depending on how much of the output you need to see.]

3. The output objects `beta`, `covbeta`, `L`, `covL`, and `Lresid` will be set to `NA` unless an HLP model is specified (i.e. `L.fct` and `X` are input).

## Author(s)

Joseph B. Lang

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

Lang, J. B. (2005) Homogeneous linear predictor models for contingency tables, *Journal of the American Statistical Association*, **100**, 121–134.

## See Also

`mph.summary`, `create.Z.ZF`, `create.U`, `num.deriv.fct`

## Examples

```
# Listed below is a collection of Basic Examples:
# https://homepage.divms.uiowa.edu/~jblang/mph.fitting/mph.basic.numerical.examples.htm

# Another collection of Less Basic Examples is online:
# https://homepage.divms.uiowa.edu/~jblang/mph.fitting/mph.numerical.examples.htm


# EXAMPLE 1. Test whether a binomial probability equals 0.5.
#
# y = (15, 22) <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# In other symbols,
#
# y = (15, 22) <- Y = (Y[1], Y[2]) ~ multinomial(37, p = (p[1], p[2])).
#
# GOAL: Test H0: p[1] = 0.5 vs. H1: not H0.

a1 <- mph.fit(y = c(15, 22), constraint = function(p) {p[1] - 0.5})

# Alternative specifications...
a2 <- mph.fit(y = c(15, 22), constraint = function(p) {p[1] - p[2]})
a3 <- mph.fit(y = c(15, 22), constraint = function(p) {log(p[1] / p[2])})
a4 <- mph.fit(y = c(15, 22), constraint = function(m) {m[1] - m[2]},
              h.mean = TRUE)
a5 <- mph.fit(y = c(15, 22), link = function(p) {p}, X = matrix(1, 2, 1))
a6 <- mph.fit(y = c(15, 22), link = "logm", X = matrix(1, 2, 1))

# Alternatively, assume that
#
# y = (15, 22) <- Y ~ MP(gamma, p | strata = 1, fixed = "none");
# i.e. Y ~ indep Poisson.
#
# In other symbols,
#
```

```
# y = (15, 22) <- Y = (Y[1], Y[2]), where
# Y[i] indep ~ Poisson(gamma * p[i]), i = 1, 2.
#
# GOAL: Test H0: p[1] = 0.5 vs. H1: not H0.

b1 <- mph.fit(y = c(15, 22), constraint = function(p) {p[1] - 0.5},
              fixed.strata = "none")

mph.summary(a1, TRUE)
mph.summary(b1, TRUE)


# EXAMPLE 2. Test whether a multinomial probability vector is uniform.
#            Test whether a multinomial probability vector equals a
#            specific value.
#
# y <- Y = (Y[1], ..., Y[6]) ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# In other symbols,
#
# y <- Y ~ multinomial(15, p = (p[1], ..., p[6]))
#
# GOAL: Test H0: p[1] = p[2] = ... = p[6] vs. H1: not H0.

y <- rmultinom(1, 15, rep(1, 6))
a1 <- mph.fit(y, L.fct = function(p) {p}, X = matrix(1, 6, 1),
              y.eps = 0.1)

# Alternative specification...
a2 <- mph.fit(y, h.fct = function(p) {as.matrix(p[-6] - 1/6)},
              y.eps = 0.1)

mph.summary(a1, TRUE)
mph.summary(a2, TRUE)

# Test whether p = (1, 2, 3, 1, 2, 3) / 12 .

p0 <- c(1, 2, 3, 1, 2, 3) / 12
b <- mph.fit(y, h.fct = function(p) {as.matrix(p[-6] - p0[-6])},
             y.eps = 0.1)
mph.summary(b, TRUE)


# EXAMPLE 3. Test whether a multinomial probability vector satisfies a
#            particular constraint.
#
# Data Source: Agresti 25:2002.
#
# y = (30, 63, 63) <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# In other symbols,
#
# y = (30, 63, 63) <- Y ~ multinomial(156, p = (p[1], p[2], p[3]))
#
# GOAL: Test H0: p[1] + p[2] = p[1] / (p[1] + p[2]) vs. H1: not H0.

y <- c(30, 63, 63)
h.fct <- function(p) {
    (p[1] + p[2]) - p[1] / (p[1] + p[2])
}
a <- mph.fit(y, h.fct = h.fct)
mph.summary(a, TRUE)


# EXAMPLE 4. Test of Independence in a 2-by-2 Table.
#
# y = (y[1, 1], y[1, 2], y[2, 1], y[2, 2]) = (25, 18, 13, 21)
#   <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# In other symbols,
# y = (y[1, 1], y[1, 2], y[2, 1], y[2, 2])
#   <- Y ~ multinomial(77, p = (p[1, 1], p[1, 2], p[2, 1], p[2, 2]))
#
# GOAL: Test H0: p[1, 1] * p[2, 2] / p[1, 2] / p[2, 1] = 1
#        vs. H1: not H0.

d <- data.frame(A = c(1, 1, 2, 2), B = c(1, 2, 1, 2),
                count = c(25, 18, 13, 21))

a1 <- mph.fit(y = d$count, h.fct = function(p)
              {log(p[1] * p[4] / p[2] / p[3])})

# Alternative specifications of independence....
a2 <- mph.fit(y = d$count, h.fct = function(p)
              {p <- matrix(p, 2, 2, byrow = TRUE);
               log(p[1, 1] * p[2, 2] / p[1, 2] / p[2, 1])})
a3 <- mph.fit(y = d$count, h.fct = function(p)
              {p[1] * p[4] / p[2] / p[3] - 1})
a4 <- mph.fit(y = d$count, h.fct = function(p)
              {p[1] / (p[1] + p[2]) - p[3] / (p[3] + p[4])})
a5 <- mph.fit(y = d$count, L.fct = "logm",
              X = model.matrix(~ A + B, data = d))

# Suppose we wished to output observed and fitted values of
# log OR, OR, and P(B = 1 | A = 1) - P(B = 1 | A = 2)...

L.fct <- function(p) {
  L <- as.matrix(c(
    log(p[1] * p[4] / p[2] / p[3]),
    p[1] * p[4] / p[2] / p[3],
    p[1] / (p[1] + p[2]) - p[3] / (p[3] + p[4])
  ))
  rownames(L) <- c("log OR", "OR",
                   "P(B = 1 | A = 1) - P(B = 1 | A = 2)")
  L
}

a6 <- mph.fit(y = d$count, h.fct = function(p)
              {log(p[1] * p[4] / p[2] / p[3])},
              L.fct = L.fct, X = diag(3))

# Unrestricted Model...
b <- mph.fit(y = d$count, L.fct = L.fct, X = diag(3))

mph.summary(a6, TRUE)
mph.summary(b, TRUE)


# EXAMPLE 5. Test of Independence in a 4-by-4 Table.
#            (Using Log-Linear Model.)
#
# Data Source: Table 2.8, Agresti, 57:2002.
#
# y <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# In other symbols,
# y <- Y ~ multinomial(96, p = (p[1, 1], p[1, 2], p[2, 1], p[2, 2]))
#
# GOAL: Test H0: p[1, 1] * p[2, 2] / p[1, 2] / p[2, 1] = 1 vs. H1: not H0.

d <- data.frame(Income = c("<15", "<15", "<15", "<15", "15-25", "15-25",
                           "15-25", "15-25", "25-40", "25-40", "25-40",
                           "25-40", ">40", ">40", ">40", ">40"),
                JobSatisf = c("VD", "LD", "MS", "VS", "VD", "LD", "MS", "VS",
                              "VD", "LD", "MS", "VS", "VD", "LD", "MS", "VS"),
                count = c(1, 3, 10, 6, 2, 3, 10, 7, 1, 6, 14, 12, 0, 1, 9, 11))

a <- mph.fit(y = d$count, link = "logp",
             X = model.matrix(~ Income + JobSatisf, data = d))
mph.summary(a)

# Alternatively,
b <- mph.fit(y = d$count, link = "logm",
             X = model.matrix(~ Income + JobSatisf, data = d))
mph.summary(b)


# EXAMPLE 6. Test Marginal Homogeneity in a 3-by-3 Table.
#
# Data Source: Table 10.16, Agresti, 445:2002.
#
# y <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# Specifically,
# y <- Y ~ multinomial(160, p = (p[1, 1], ..., p[3, 3]))
#
# GOAL: Test H0: p[1, +] = p[+, 1], p[2, +] = p[+, 2], p[3, +] = p[+, 3]
#       vs. H1: not H0.

d <- data.frame(Siskel = c("Pro", "Pro", "Pro", "Mixed", "Mixed",
                           "Mixed", "Con", "Con", "Con"),
                Ebert = c("Pro", "Mixed", "Con", "Pro", "Mixed",
                          "Con", "Pro", "Mixed", "Con"),
                count = c(64, 9, 10, 11, 13, 8, 13, 8, 24))

h.fct <- function(p){
    p.Siskel <- M.fct(d$Siskel) %*% p
    p.Ebert  <- M.fct(d$Ebert) %*% p
    as.matrix(c(p.Siskel[-3] - p.Ebert[-3]))
}
a1 <- mph.fit(y = d$count, h.fct = h.fct)
mph.summary(a1, TRUE)

# Suppose that we wish to report on the observed and fitted
# marginal probabilities.

L.fct <- function(p) {
    p.Siskel <- M.fct(d$Siskel) %*% p
    p.Ebert <- M.fct(d$Ebert) %*% p
    L <- as.matrix(c(p.Siskel, p.Ebert))
    rownames(L) <- c(paste(sep = "", "P(Siskel=", levels(as.factor(d$Siskel)), ")"),
                     paste(sep = "", "P(Ebert=", levels(as.factor(d$Ebert)), ")"))
    L
}
a2 <- mph.fit(y = d$count, h.fct = h.fct, L.fct = L.fct, X = diag(6))
mph.summary(a2, TRUE)

# M.fct(factor) %*% p gives the marginal probabilities corresponding to
# the levels of 'factor'. The marginal probabilities are ordered by the
# levels of 'factor'.
#
# Alternatively, in this rectangular table setting, we can find the
# marginal probabilities using the apply(...) function. In this case,
# the marginal probabilities are ordered as they are entered in the
# data set.

h.fct <- function(p) {
    p <- matrix(p, 3, 3, byrow = TRUE)
    p.Siskel <- apply(p, 1, sum)
    p.Ebert <- apply(p, 2, sum)
    as.matrix(c(p.Siskel[-3] - p.Ebert[-3]))
}

L.fct <- function(p) {
    p <- matrix(p, 3, 3, byrow = TRUE)
    p.Siskel <- apply(p, 1, sum)
    p.Ebert <- apply(p, 2, sum)
    L <- as.matrix(c(p.Siskel, p.Ebert))
    rownames(L) <- c("P(Siskel=Pro)", "P(Siskel=Mixed)",
                     "P(Siskel=Con)", "P(Ebert=Pro)",
                     "P(Ebert=Mixed)", "P(Ebert=Con)")
    L
}
b <- mph.fit(y = d$count, h.fct = h.fct, L.fct = L.fct, X = diag(6))


# EXAMPLE 7. Log-Linear Model for 2-by-2-by-2 Table.
#
# Data Source: Table 8.16, Agresti 347:2002
#
# y <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# Specifically,
#
# y <- Y ~ multinomial(621, p).
#
# The counts in y are cross-classification counts for variables
# G = Gender, I = Information Opinion, H = Health Opinion.
#
# GOAL: Fit the loglinear models [GI, GH, IH] and [G, IH].

d <- data.frame(G = c("Male", "Male", "Male", "Male",
                      "Female", "Female", "Female", "Female"),
                I = c("Support", "Support", "Oppose", "Oppose",
                      "Support", "Support", "Oppose", "Oppose"),
                H = c("Support", "Oppose", "Support", "Oppose",
                      "Support", "Oppose", "Support", "Oppose"),
                count = c(76, 160, 6, 25, 114, 181, 11, 48))

# Fit loglinear model [GI, GH, IH]...

a1 <- mph.fit(y = d$count, link = "logm",
              X = model.matrix(~ G + I + H + G:I + G:H + I:H, data = d))

# Fit loglinear model [G, IH]...

a2 <- mph.fit(y = d$count, link = "logm",
              X = model.matrix(~ G + I + H + I:H, data = d))

# Different Sampling Distribution Assumptions:
#
# Alternatively, assume
# y <- Y ~ MP(gamma, p | strata = 1, fixed = "none");
# that is, Y ~ indep Poisson.
#
# In other symbols,
# y <- Y, where Y[i] indep ~ Poisson(m[i] = gamma * p[i]).
# Here, gamma is the unknown expected sample size.
```

```
b2 <- mph.fit(y = d$count, link = "logm",
              X = model.matrix(~ G + I + H + I:H, data = d),
              fixed = "none")

# Alternatively, assume
# y <- Y ~ MP(gamma, p | strata = Gender, fixed = "all");
# that is, Y ~ prod multinomial.
#
# In other symbols,
# y <- Y = (Y[1, 1, 1], Y[1, 1, 2], ..., Y[2, 2, 2]),
# where (Y[i, 1, 1], ..., Y[i, 2, 2]) indep ~ multinomial(n[i], p[i, , ]).
# Here, p[i, j, k] = P(I = j, H = k | G = i) and n[1] = 267 and
# n[2] = 354 are the a priori fixed sample sizes for males and females.

c2 <- mph.fit(y = d$count, link = "logm",
              X = model.matrix(~ G + I + H + I:H, data = d),
              strata = d$G)

# Alternatively, assume
# y <- Y ~ MP(gamma, p | strata = Gender, fixed = "none");
# that is, Y ~ prod Poisson.
#
# In other symbols,
# y <- Y = (Y[1, 1, 1], Y[1, 1, 2], ..., Y[2, 2, 2]),
# where Y[i, j, k] indep ~ Poisson(m[i, j, k] = gamma[i] * p[i, j, k]).
# Here, p[i, j, k] = P(I = j, H = k | G = i) and gamma[1] and gamma[2] are the
# unknown expected sample sizes for males and for females.

d2 <- mph.fit(y = d$count, link = "logm",
              X = model.matrix(~ G + I + H + I:H, data = d),
              strata = d$G, fixed = "none")

cbind(a2$m, b2$m, c2$m, d2$m, sqrt(diag(a2$covm)), sqrt(diag(b2$covm)),
      sqrt(diag(c2$covm)), sqrt(diag(d2$covm)))
cbind(a2$p, b2$p, c2$p, d2$p, sqrt(diag(a2$covp)), sqrt(diag(b2$covp)),
      sqrt(diag(c2$covp)), sqrt(diag(d2$covp)))


# EXAMPLE 8. Fit Linear-by-Linear Log-Linear Model
#
# Data Source: Table 8.15, Agresti, 345:2002
#
# y <- Y ~ MP(gamma, p | strata = 1, fixed = "all");
# i.e. Y ~ multinomial.
#
# Specifically,
# y <- Y ~ multinomial(1425, p)
#
# GOAL: Assess the fit of the linear-by-linear log-linear model.

d <- list(Schooling = c("<HS", "<HS", "<HS", "HS", "HS", "HS", ">HS", ">HS", ">HS"),
          Abortion = c("Disapprove", "Middle", "Approve", "Disapprove", "Middle",
                       "Approve", "Disapprove", "Middle", "Approve"),
                count = c(209, 101, 237, 151, 126, 426, 16, 21, 138))

Schooling.score <- -1 * (d$Schooling == "<HS") +
                    0 * (d$Schooling == "HS") +
                    1 * (d$Schooling == ">HS")
Abortion.score  <- -1 * (d$Abortion == "Disapprove") +
                    0 * (d$Abortion == "Middle") +
                    1 * (d$Abortion == "Approve")

d <- data.frame(d, Schooling.score, Abortion.score)

a <- mph.fit(y = d$count, link = "logm",
             X = model.matrix(~ Schooling + Abortion +
             Schooling.score : Abortion.score, data = d))
mph.summary(a, TRUE)


# EXAMPLE 9. Marginal Standardization of a Contingency Table.
#
# Data Source: Table 8.15, Agresti 345:2002.
#
# GOAL: For a two-way table, find the standardized values of y, say y*,
# that satisfy (i) y* has the same odds ratios as y, and
#             (ii) y* has row and column totals equal to 100.
#
# Note: This is equivalent to the problem of finding the fitted values
# for the following model...
# x <- Y ~ multinomial(n, p = (p[1, 1], ..., p[3, 3]))
#      p[1, +] = p[2, +] = p[3, +] = p[+, 1] = p[+, 2] = p[+, 3] = 1/3
#      p[1, 1] * p[2, 2] / p[2, 1] / p[1, 2] = or[1, 1]
#      p[1, 2] * p[2, 3] / p[2, 2] / p[1, 3] = or[1, 2]
#      p[2, 1] * p[3, 2] / p[3, 1] / p[2, 2] = or[2, 1]
#      p[2, 2] * p[3, 3] / p[3, 2] / p[2, 3] = or[2, 2],
# where or[i, j] = y[i, j] * y[i + 1, j + 1] / y[i + 1, j] / y[i, j + 1]
# are the observed (y) odds ratios.
# If m is the vector of fitted values, then y* = m * 300 / sum(m)
# are the standardized values of y.
# Here x can be any vector of 9 counts.
# Choosing x so that the sum is 300 leads to sum(m) = 300, so that
# y* = m in this case.

d <- data.frame(Schooling = c("<HS", "<HS", "<HS", "HS", "HS", "HS", ">HS", ">HS", ">HS"),
                Abortion = c("Disapprove", "Middle", "Approve", "Disapprove", "Middle",
                             "Approve", "Disapprove", "Middle", "Approve"),
                count = c(209, 101, 237, 151, 126, 426, 16, 21, 138))

h.fct <- function(p) {
   p.Schooling <- M.fct(d$Schooling) %*% p
   p.Abortion  <- M.fct(d$Abortion) %*% p
   p <- matrix(p, 3, 3, byrow = TRUE)
   as.matrix(c(
     p.Schooling[-3] - 1/3, p.Abortion[-3] - 1/3,
     p[1, 1] * p[2, 2] / p[2, 1] / p[1, 2] - 209 * 126 / 151 / 101,
      p[1, 2] * p[2, 3] / p[2, 2] / p[1, 3] - 101 * 426 / 126 / 237,
      p[2, 1] * p[3, 2] / p[3, 1] / p[2, 2] - 151 * 21 / 16 / 126,
      p[2, 2] * p[3, 3] / p[3, 2] / p[2, 3] - 126 * 138 / 21 / 426
   ))
}

b <- mph.fit(y = d$count, h.fct = h.fct)
ystar <- b$m * 300 / sum(b$m)
matrix(round(ystar, 1), 3, 3, byrow = TRUE)

x <- c(rep(33, 8), 36)
b <- mph.fit(y = x, h.fct = h.fct)
ystar <- b$m
matrix(round(ystar, 1), 3, 3, byrow = TRUE)


# EXAMPLE 10. Cumulative Logit Model.
#
# Data Source: Table 7.19, Agresti, 306:2002.
#
# y <- Y ~ MP(gamma, p | strata = Therapy * Gender, fixed = "all");
# i.e. Y ~ prod multinomial.
#
# Here, y[i, j, k] is the cross-classification count corresponding to
# Therapy = i, Gender = j, Response = k.
#
# The table probabilities are defined as
# p[i, j, k] = P(Response = k | Therapy = i, Gender = j).
#
# Goal: Fit the cumulative logit proportional odds model that includes
# the main effect of Therapy and Gender.

d <- data.frame(Therapy = c("Sequential", "Sequential", "Sequential", "Sequential",
                            "Sequential", "Sequential", "Sequential", "Sequential",
                            "Alternating", "Alternating", "Alternating", "Alternating",
                            "Alternating", "Alternating", "Alternating", "Alternating"),
                Gender = c("Male", "Male", "Male", "Male", "Female", "Female",
                           "Female", "Female", "Male", "Male", "Male", "Male",
                           "Female", "Female", "Female", "Female"),
                Response = c("Progressive", "NoChange", "Partial", "Complete",
                             "Progressive", "NoChange", "Partial", "Complete",
                             "Progressive", "NoChange", "Partial", "Complete",
                             "Progressive", "NoChange", "Partial", "Complete"),
                count = c(28, 45, 29, 26, 4, 12, 5, 2, 41, 44, 20, 20, 12, 7, 3, 1))

strata <- paste(sep = "", d$Therapy, ".", d$Gender)
d <- data.frame(d, strata)

d3 <- subset(d, Response != "Complete")
levels(d3$Response) <- c(NA, "NoChange", "Partial", "Progressive")

L.fct <- function(p) {
   p <- matrix(p, 4, 4, byrow = TRUE)
```

```
    clogit <- c()
    for (s in 1:4) {
      clogit <- c(clogit,
                  log(sum(p[s, 1])   / sum(p[s, 2:4])),
                  log(sum(p[s, 1:2]) / sum(p[s, 3:4])),
                  log(sum(p[s, 1:3]) / sum(p[s, 4]))
      )
    }
    L <- as.matrix(clogit)
    rownames(L) <- c(paste(sep = "", "log odds(R < ", 2:4, "|",
                           d3$strata, ")"))
    L
}

a <- mph.fit(d$count, link = L.fct,
             X = model.matrix(~ -1 + Response + Therapy + Gender,
                              data = d3),
             strata = strata)

# Fit the related non-proportional odds cumulative logit model
b <- mph.fit(d$count, link = L.fct,
             X = model.matrix(~ Response + Response * Therapy +
                                Response * Gender - 1 - Therapy - Gender,
                              data = d3),
             strata = strata)

mph.summary(a, TRUE)
mph.summary(b, TRUE)
```

## `mph.summary` *Summary Statistics of the Fitted MPH Model*

### Description

Computes and prints a collection of summary statistics of the fitted MPH model.

This function is used in conjunction with the ML fitting function `mph.fit`.

### Usage

```
mph.summary(mph.out, cell.stats = FALSE, model.info = FALSE, digits = 4)
```

### Arguments

| | |
|---|---|
| `mph.out` | Result of `mph.fit`. |
| `cell.stats` | Logical variable indicating whether cell specific statistics are to be output. Default: `cell.stats` = `FALSE`. |
| `model.info` | Logical variable indicating whether model information is to be output. Default: `model.info` = `FALSE`. |
| `digits` | Integer giving output precision; used in the `round()` function. |

## Value

NULL

## Author(s)

Joseph B. Lang

## References

Lang, J. B. (2004) Multinomial-Poisson homogeneous models for contingency tables, *Annals of Statistics*, **32**, 340–383.

Lang, J. B. (2005) Homogeneous linear predictor models for contingency tables, *Journal of the American Statistical Association*, **100**, 121–134.

## See Also

mph.fit

---

nested_Gsq_nr *Nested G-Squared Statistic Based CIs (Non-Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the nested $G^2$ statistic, for estimands in contingency tables subject to equality constraints.

The program may stop because of a non-convergence issue.

## Usage

```
nested_Gsq_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv, S0.fct,
              S0.fct.deriv, max.mph.iter, step, change.step.after,
              y.eps, iter.orig, norm.diff.conv, norm.score.conv,
              max.score.diff.iter, S.space.H0, tol.psi, tol,
              max.iter, cut.off, delta)
```

## Arguments

| | |
|---|---|
| y | Observed table counts in the contingency table(s), in vector form. |
| strata | Vector of the same length as y that gives the stratum membership identifier. |
| fixed.strata | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| h0.fct | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| h0.fct.deriv | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If h0.fct.deriv is not specified or h0.fct.deriv = NULL, numerical derivatives will be used. |

| | |
|---|---|
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |

## Value

Provided that `nested_Gsq_nr` does not stop, it returns a 1-by-2 matrix which displays two endpoints of the confidence interval based on the nested $G^2$ statistic.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`nested_Gsq_nr`, `f.psi`, `ci.table`

---

## `nested_Gsq_robust` *Nested G-Squared Statistic Based CIs (Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the nested $G^2$ statistic, for estimands in contingency tables subject to equality constraints.

## Usage

```
nested_Gsq_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                  S0.fct, S0.fct.deriv, max.mph.iter, step,
                  change.step.after, y.eps, iter.orig, norm.diff.conv,
                  norm.score.conv, max.score.diff.iter, S.space.H0,
                  tol.psi, tol, max.iter, cut.off, delta, adj.epsilon,
                  iter.robust.max, iter.robust.eff)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `adj.epsilon`, `iter.robust.max`, `iter.robust.eff` | The parameters used in the robustifying procedure. |

## Value

`nested_Gsq_robust` returns a list, which includes two objects. The first object is a 1-by-2 matrix which displays two endpoints of the confidence interval based on the nested $G^2$ statistic. For the second object, it includes the warning message that occurs during construction of the confidence interval if the robustifying procedure is evoked: `"nested.Gsq.CI: Adjustment used. Not on original data.\n"`. If the robustifying procedure is not evoked, the second object is `NULL`.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`nested_Gsq_nr`, `f.psi`, `ci.table`

---

## `nested_PD_nr` *Nested Power-Divergence Statistic Based CIs (Non-Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the nested power-divergence statistic, for estimands in contingency tables subject to equality constraints.

The program may stop because of a non-convergence issue.

## Usage

```
nested_PD_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv, S0.fct,
             S0.fct.deriv, max.mph.iter, step, change.step.after,
             y.eps, iter.orig, norm.diff.conv, norm.score.conv,
             max.score.diff.iter, S.space.H0, tol.psi, tol,
             max.iter, cut.off, delta, pdlambda)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv` = `NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv` = `NULL`, numerical derivatives will be used. |

| | |
|---|---|
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. |

## Value

Provided that `nested_PD_nr` does not stop, it returns a 1-by-2 matrix which displays two endpoints of the confidence interval based on the nested power-divergence statistic.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`nested_PD_robust`, `f.psi`, `ci.table`

---

## `nested_PD_robust` *Nested Power-Divergence Statistic Based CIs (Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the nested power-divergence statistic, for estimands in contingency tables subject to equality constraints.

## Usage

```
nested_PD_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                 S0.fct, S0.fct.deriv, max.mph.iter, step,
                 change.step.after, y.eps, iter.orig, norm.diff.conv,
                 norm.score.conv, max.score.diff.iter, S.space.H0,
                 tol.psi, tol, max.iter, cut.off, delta, pdlambda,
                 adj.epsilon, iter.robust.max, iter.robust.eff)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `pdlambda` | The index parameter $\lambda$ in the power-divergence statistic. |
| `adj.epsilon`, `iter.robust.max`, `iter.robust.eff` | The parameters used in the robustifying procedure. |

## Value

`nested_PD_robust` returns a list, which includes two objects. The first object is a 1-by-2 matrix which displays two endpoints of the confidence interval based on the nested power-divergence statistic. For the second object, it includes the warning message that occurs during construction of the confidence interval if the robustifying procedure is evoked: `"nested.PD.CI: Adjustment used. Not on original data.\n"`. If the robustifying procedure is not evoked, the second object is `NULL`.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`nested_PD_nr`, `f.psi`, `ci.table`

---

`nested_Xsq_nr` *Nested $X$-Squared Statistic Based CIs (Non-Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the nested $X^2$ statistic, for estimands in contingency tables subject to equality constraints.

The program may stop because of a non-convergence issue.

## Usage

```
nested_Xsq_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv, S0.fct,
              S0.fct.deriv, max.mph.iter, step, change.step.after,
              y.eps, iter.orig, norm.diff.conv, norm.score.conv,
              max.score.diff.iter, S.space.H0, tol.psi, tol,
              max.iter, cut.off, delta)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |

| | |
|---|---|
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |

## Value

Provided that `nested_Xsq_nr` does not stop, it returns a 1-by-2 matrix which displays two endpoints of the confidence interval based on the nested $X^2$ statistic.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`nested_Xsq_nr`, `f.psi`, `ci.table`

---

## `nested_Xsq_robust` *Nested $X$-Squared Statistic Based CIs (Robust)*

---

## Description

Constructs confidence intervals (CIs), based on the nested $X^2$ statistic, for estimands in contingency tables subject to equality constraints.

## Usage

```
nested_Xsq_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                  S0.fct, S0.fct.deriv, max.mph.iter, step,
                  change.step.after, y.eps, iter.orig, norm.diff.conv,
                  norm.score.conv, max.score.diff.iter, S.space.H0,
                  tol.psi, tol, max.iter, cut.off, delta, adj.epsilon,
                  iter.robust.max, iter.robust.eff)
```

## Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `tol.psi`, `tol`, `max.iter` | The parameters used in the three stopping criteria in solving for the roots to the test-inversion equation. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `delta` | The constant $\delta$ that is in expressions of the moving critical values within each sliding quadratic step. |
| `adj.epsilon`, `iter.robust.max`, `iter.robust.eff` | The parameters used in the robustifying procedure. |

## Value

`nested_Xsq_robust` returns a list, which includes two objects. The first object is a 1-by-2 matrix which displays two endpoints of the confidence interval based on the nested $X^2$ statistic. For the second object, it includes the warning message that occurs during construction of the confidence interval if the robustifying procedure is evoked: `"nested.Xsq.CI: Adjustment used. Not on original data.\n"`. If the robustifying procedure is not evoked, the second object is `NULL`.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

nested_Xsq_nr, f.psi, ci.table

---

`num.deriv.fct` *Numerical Derivatives Based on Central Difference Formula*

---

## Description

Computes the numerical derivative of the transpose of the vector-valued function $f$ evaluated at the point $m$, based on the central difference formula.

If $f$ is a mapping from $R^p$ to $R^q$, then the result is a $p$-by-$q$ matrix. i.e. The result is an approximation to $\partial f'(m)/\partial m$.

## Usage

```
num.deriv.fct(f.fct, m)
```

## Arguments

| | |
|---|---|
| f.fct | An R function object that defines a vector-valued function $f$. |
| m | A vector, indicating the point $m$ at which the numerical derivative is to be computed. |

## Value

num.deriv.fct returns a matrix, which is the numerical derivative of the transpose of the function $f$ evaluated at $m$.

## Author(s)

Joseph B. Lang

## Examples

```
# Let x = (x[1], x[2], x[3])', and
# f(x) = (x[1]^3 - 2 * x[2] + 1, sin(x[1] * x[3]), log(x[2] + x[3]))'.
# Approximate d f^{T}(x) / d x  at x = (1, 2, 3)'.
# The true value of the derivative is
# [ 3   3cos(3)    0
#  -2      0      0.2
#   0    cos(3)   0.2] .

f.fct <- function(x) {
  c(x[1]^3 - 2 * x[2] + 1,
    sin(x[1] * x[3]),
    log(x[2] + x[3]))
}
num.deriv.fct(f.fct, c(1, 2, 3))
```

## `quadratic.fit` *Quadratic Fit*

### Description

Fits a quadratic curve that passes all three points on the two-dimensional Euclidean space $R^2$.

If the design matrix $X$ of the quadratic fit has a condition number which is greater than $10^8$, a linear regression line is fitted to the three points instead.

### Usage

```
quadratic.fit(x, y)
```

### Arguments

| | |
|---|---|
| `x` | A vector of length three, which represents the $x$-values of the three points. |
| `y` | A vector of length three, which represents the $y$-values of the three points. |

### Value

`quadratic.fit` returns a vector of length three. The first, second, and third elements of the returned vector are the second degree, first degree, and zero-th degree coefficients, respectively, of the fitted quadratic curve, or of the fitted linear regression line.

### Author(s)

Qiansheng Zhu

### See Also

`solve_quadratic`

### Examples

```
# Three points: (0, 1), (1, 0), (3, 4).
quadratic.fit(c(0, 1, 3), c(1, 0, 4))
```

## `solve_quadratic` *Solve for Real Root(s) to the Quadratic Equation*

### Description

Solves for real-valued roots to the quadratic equation $ax^2 + bx + c = 0$.

### Usage

```
solve_quadratic(a, b, c)
```

### Arguments

| | |
|---|---|
| a, b, c | Coefficients in the quadratic equation $ax^2 + bx + c = 0$. |

### Value

`solve_quadratic` returns a list, which includes the following two objects:

| | |
|---|---|
| `flag` | Indicates the number of distinct real roots to the quadratic equation. It can be one of `"infinite"`, `"none"`, `"one"`, or `"two"`. |
| `x` | Real root(s) to the quadratic equation. If `flag = "infinite"`, we simply write `x = 0`; If `flag = "none"`, we write `x = NA`. |

### Author(s)

Qiansheng Zhu

### See Also

`quadratic.fit`

### Examples

```
solve_quadratic(1, 2, 1)
solve_quadratic(1, 2, 2)
solve_quadratic(0, 2, 1)
```

## `Wald_trans.Wald_nr` *Wald-Type CIs (Non-Robust)*

### Description

Constructs non-transformed and transformed (if the transformation $g$ is specified) Wald confidence intervals (CIs) for estimands in contingency tables subject to equality constraints.

The program may stop because of a non-convergence issue.

### Usage

```
Wald_trans.Wald_nr(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                   S0.fct, S0.fct.deriv, max.mph.iter, step,
                   change.step.after, y.eps, iter.orig, norm.diff.conv,
                   norm.score.conv, max.score.diff.iter, cut.off,
                   S.space.H0, trans.g, trans.g.deriv, trans.g.inv)
```

### Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `trans.g` | The transformation $g$ used in the transformed Wald confidence interval. |
| `trans.g.deriv` | The derivative function of the transformation $g$, i.e. $dg(w)/dw$. If it is specified, it should be an R function, even if the derivative function is a constant function. |
| `trans.g.inv` | $g^{-1}$ function used in back-transformation step in construction of the transformed Wald confidence interval. |

### Value

Provided that `Wald_trans.Wald_nr` does not stop,

- either it returns a 1-by-2 matrix which displays two endpoints of the non-transformed Wald confidence interval, if the transformation $g$ is not specified;
- or it returns a 2-by-2 matrix, whose first row displays two endpoints of the non-transformed Wald confidence interval, and whose second row displays two endpoints of the transformed Wald confidence interval, if the transformation $g$ is specified.

### Author(s)

Qiansheng Zhu

### References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

### See Also

`Wald_trans.Wald_robust`, `f.psi`, `ci.table`

---

`Wald_trans.Wald_robust`

## *Wald-Type CIs (Robust)*

---

### Description

Constructs non-transformed and transformed (if the transformation $g$ is specified) Wald confidence intervals (CIs) for estimands in contingency tables subject to equality constraints.

### Usage

```
Wald_trans.Wald_robust(y, strata, fixed.strata, h0.fct, h0.fct.deriv,
                       S0.fct, S0.fct.deriv, max.mph.iter, step,
                       change.step.after, y.eps, iter.orig, norm.diff.conv,
                       norm.score.conv, max.score.diff.iter, cut.off,
                       S.space.H0, trans.g, trans.g.deriv, trans.g.inv,
                       adj.epsilon, iter.robust.max, iter.robust.eff)
```

### Arguments

| | |
|---|---|
| `y` | Observed table counts in the contingency table(s), in vector form. |
| `strata` | Vector of the same length as `y` that gives the stratum membership identifier. |
| `fixed.strata` | The object that gives information on which stratum (strata) has (have) fixed sample sizes. |

| | |
|---|---|
| `h0.fct` | The constraint function $h_0(\cdot)$ with respect to $m$, where $m = E(Y)$, the vector of expected table counts. |
| `h0.fct.deriv` | The R function object that computes analytic derivative of the transpose of the constraint function $h_0(\cdot)$ with respect to $m$. If `h0.fct.deriv` is not specified or `h0.fct.deriv = NULL`, numerical derivatives will be used. |
| `S0.fct` | The estimand function $S_0(\cdot)$ with respect to $m$. |
| `S0.fct.deriv` | The R function object that computes analytic derivative of the estimand function $S_0(\cdot)$ with respect to $m$. If `S0.fct.deriv` is not specified or `S0.fct.deriv = NULL`, numerical derivatives will be used. |
| `max.mph.iter`, `step`, `change.step.after`, `y.eps`, `iter.orig`, `norm.diff.conv`, `norm.score.conv`, `max.score.diff.iter` | The parameters used in `mph.fit`. |
| `cut.off` | `qchisq(cc, 1)`. i.e. The chi-square cutoff, with 1 df, based on the significance level `1-cc`. |
| `S.space.H0` | Restricted estimand space of $S(\cdot)$ under $H_0$, i.e. subject to the imposed equality constraints along with sampling constraints. |
| `trans.g` | The transformation $g$ used in the transformed Wald confidence interval. |
| `trans.g.deriv` | The derivative function of the transformation $g$, i.e. $dg(w)/dw$. If it is specified, it should be an R function, even if the derivative function is a constant function. |
| `trans.g.inv` | $g^{-1}$ function used in back-transformation step in construction of the transformed Wald confidence interval. |
| `adj.epsilon`, `iter.robust.max`, `iter.robust.eff` | The parameters used in the robustifying procedure. |

## Value

`Wald_trans.Wald_robust` returns a list, which includes two objects. The first object is

- either a 1-by-2 matrix which displays two endpoints of the non-transformed Wald confidence interval, if the transformation $g$ is not specified;
- or a 2-by-2 matrix, whose first row displays two endpoints of the non-transformed Wald confidence interval, and whose second row displays two endpoints of the transformed Wald confidence interval, if the transformation $g$ is specified.

For the second object, it includes the warning message that occurs during construction of the confidence interval(s) if the robustifying procedure is evoked: `"Wald.CI: Adjustment used. Not on original data.\n"`, or `"Wald.CI and trans.Wald.CI: Adjustment used. Not on original data.\n"`. If the robustifying procedure is not evoked, the second object is `NULL`.

## Author(s)

Qiansheng Zhu

## References

Zhu, Q. (2020) "On improved confidence intervals for parameters of discrete distributions." PhD dissertation, University of Iowa.

## See Also

`Wald_trans.Wald_nr`, `f.psi`, `ci.table`

# Index