Package: ordinalTables (via r-universe)

May 16, 2026

Type Package

Title Fit Models to Two-Way Tables with Correlated Ordered Response Categories

Version 1.0.0.3

Description Fit a variety of models to two-way tables with ordered categories. Most of the models are appropriate to apply to tables of that have correlated ordered response categories. There is a particular interest in rater data and models for rescore tables. Some utility functions (e.g., Cohen's kappa and weighted kappa) support more general work on rater agreement. Because the names of the models are very similar, the functions that implement them are organized by last name of the primary author of the article or book that suggested the model, with the name of the function beginning with that author's name and an underscore. This may make some models more difficult to locate if one doesn't have the original sources. The vignettes and tests can help to locate models of interest. For more dertaiils see the following references: Agresti, A. (1983) <doi:10.1016/0167-7152(83)90051-2> ``A Simple Diagonals-Parameter Symmetry And Quasi-Symmetry Model'', Agrestim A. (1983) <doi:10.2307/2531022> ``Testing Marginal Homogeneity for Ordinal Categorical Variables'', Agresti, A. (1988) <doi:10.2307/2531866> ``A Model For Agreement Between Ratings On An Ordinal Scale'', Agresti, A. (1989) <doi:10.1016/0167-7152(89)90104-1> ``An Agreement Model With Kappa As Parameter'', Agresti, A. (2010 ISBN:978-0470082898) ``Analysis Of Ordinal Categorical Data'', Bhapkar, V. P. (1966) <doi:10.1080/01621459.1966.10502021> ``A Note On The Equivalence Of Two Test Criteria For Hypotheses In Categorical Data'', Bhapkar, V. P. (1979) <doi:10.2307/2530344> ``On Tests Of Marginal Symmetry And Quasi-Symmetry In Two And Three-Dimensional Contingency Tables'', Bowker, A. H. (1948) <doi:10.2307/2280710> ``A Test For Symmetry In Contingency Tables'', Clayton, D. G. (1974) <doi:10.2307/2335638> ``Some Odds Ratio Statistics For The Analysis Of Ordered Categorical Data'', Cliff, N. (1993) <doi:10.1037/0033-2909.114.3.494> ``Dominance Statistics: Ordinal Analyses To Answer Ordinal Questions", Cliff, N. (1996 ISBN:978-0805813333) ``Ordinal Methods For Behavioral Data Analysis", Goodman, L. A. (1979) <doi:10.1080/01621459.1979.10481650> ``Simple Models For The Analysis Of Association In Cross-Classifications Having Ordered Categories", Goodman, L. A. (1979) <doi:10.2307/2335159> ``Multiplicative Models For Square Contingency Tables With Ordered Categories", Ireland, C. T., Ku, H. H., & Kullback, S. (1969) <doi:10.2307/2286071> ``Symmetry And Marginal Homogeneity Of An $r \times r$ Contingency Table", Ishi-kuntz, M. (1994 ISBN:978-0803943766) ``Ordinal Log-linear Models", McCullah, P. (1977) <doi:10.2307/2345320> ``A Logistic Model For Paired Comparisons With Ordered Categorical Data", McCullagh, P. (1978) <doi:10.2307/2335224> A Class Of Parametric Models For The Analysis Of Square Contingency Tables With Ordered Categories``, McCullagh, P. (1980) <doi:10.1111/j.2517-6161.1980.tb01109.x> "Regression Models For Ordinal Data``, Penn State: Eberly College of Science (undated) <https://online.stat.psu.edu/stat504/lesson/11> "Stat 504: Analysis of Discrete Data, 11. Advanced Topics I``, Schuster, C. (2001) <doi:10.3102/10769986026003331> "Kappa As A Parameter Of A Symmetry Model For Rater Agreement``, Shoukri, M. M. (2004 ISBN:978-1584883210). "Measures Of Interobserver Agreement``, Stuart, A. (1953) <doi:10.2307/2333101> "The Estimation Of And Comparison Of Strengths Of Association In Contingency Tables``, Stuart, A. (1955) <doi:10.2307/2333387> "A Test For Homogeneity Of The Marginal Distributions In A Two-Way Classification``, von Eye, A., & Mun, E. Y. (2005 ISBN:978-0805849677) "Analyzing Rater Agreement: Manifest Variable Methods".

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports MASS

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat

Config/testthat/edition 3

Depends R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation no

Author John R. Donoghue [aut, cre]

Maintainer John R. Donoghue <jdonoghue0823@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2025-09-18 08:00:02 UTC

RemoteUrl https://github.com/cran/ordinalTables

RemoteRef HEAD

RemoteSha d65d652ee71a11249ba101f8fde2810c04e2656b

Contents

`Agresti_bisection` *Solves equation Agresti_f() = 0 for delta by method of bisection..*

Description

Solves equation Agresti_f() = 0 for delta by method of bisection..

Usage

```
Agresti_bisection(p, pi_margin, x_low = 0, x_high = 1)
```

Arguments

`p`	matrix of observed proportions
`pi_margin`	current value of (row and column) marginal proportion
`x_low`	lower bound for search. Default value is 0.0
`x_high`	upper bound for search. Default value is 1.0

Value

value of kappa that makes the function 0.0

`Agresti_compute_lambda` *Computes value of lambda parameter*

Description

Computes value of lambda parameter

Usage

```
Agresti_compute_lambda(p, pi)
```

Arguments

`p`	matrix of observed proportions
`pi`	matrix of model-supplied proportions

Value

value of the lambda parameter

`Agresti_compute_pi` *Computes the matrix pi of model-based proportions*

Description

Computes the matrix pi of model-based proportions

Usage

```
Agresti_compute_pi(pi_margin, kappa)
```

Arguments

`pi_margin`	current value of (row and column) marginal proportion
`kappa`	current estimate of kappa coefficient

Value

matrix of model-based proportions

`Agresti_create_design_matrix` *Creates the design matrix for Agresti's simple diagonal quasi-symmetry model.*

Description

This parameterization does not match equation (2.2) in the paper, but it yields results that are identical to those in the paper. Agresti, A. (1983), A simple diagonals-parameter symmetry and quasi-symmetry model. Statistics and Probability Letters I, 313-316.

Usage

```
Agresti_create_design_matrix(n_dim)
```

Arguments

`n_dim`	the size of the date matrix

Value

the design matrix for the model, that can bee used with ml_for_log_linear

---

`Agresti_equation_1` *First equation in section 3. Solved for kappa.*

Description

First equation in section 3. Solved for kappa.

Usage

```
Agresti_equation_1(p, pi_margin, kappa)
```

Arguments

`p`	matrix of observed proportions
`pi_margin`	current value of (row and column) marginal proportion
`kappa`	current value of coefficient kappa

`Agresti_equation_2` *Second equation in section 3. Solved for pi_margin.*

Description

Second equation in section 3. Solved for pi_margin.

Usage

```
Agresti_equation_2(p, pi_margin, lambda, kappa)
```

Arguments

`p`	matrix of observed proportions
`pi_margin`	current value of (row and column) marginal proportion
`lambda`	value of quantity lambda defined in third equation
`kappa`	current value of coefficient kappa

Agresti_equation_3 *Third equation in section 3. Solved for lambda*

Description

Third equation in section 3. Solved for lambda

Usage

```
Agresti_equation_3(p, pi_margin, kappa)
```

Arguments

p	matrix of observed proportions
pi_margin	current value of (row and column) marginal proportion
kappa	current valye of coefficient kappa

Agresti_extract_delta *Extracts the quasi-symmetry information from the result provided.*

Description

Extracts the quasi-symmetry information from the result provided.

Usage

```
Agresti_extract_delta(result)
```

Arguments

result	result of call to log_linear_fit()

Value

list consisting of beta: the beta coefficient se: the standard error of beta z: the ratio beta / se delta: the delta coefficient = exp(2.0 * beta)

Agresti_f — *Function value for first equation in section 3.*

Description

Used by Agresti_bisection()

Usage

```
Agresti_f(p, pi_margin, kappa)
```

Arguments

p	matrix of observed proportions
pi_margin	current value of (row and column) marginal proportion
kappa	current estimate of kappa coefficient

Agresti_kappa_agreement — *Fits Agresti's agreement model that includes kappa as a parameter.*

Description

Agresti, A. (1989). An agreement model with kappa as a parameter. Statistics and Probability Letters, 7, 271-273.

Usage

```
Agresti_kappa_agreement(n, verbose = FALSE)
```

Arguments

n	matrix of observed counts
verbose	should cycle-by-cycle info be printed as messages? The default is FALSE.

Value

a list containing kappa: value of kappa coefficient pi_margin: value of marginal p-values. They apply to rows and columns chisq: Pearson X^2 df: degrees of freedom expected: fitted frequencies

Agresti_simple_diagonals_parameter_quasi_symmetry

Agresti's simple diganal quasi-symmetry model.

Description

This parameterization does not match equation (2.2) in the paper, but it yields results that are identical to those in the paper. Agresti, A. (1983), A simple diagonals-parameter symmetry and quasi-symmetry model. Statistics and Probability Letters I, 313-316.

Usage

```
Agresti_simple_diagonals_parameter_quasi_symmetry(n)
```

Arguments

n	the matrix of observed counts

Value

a list containing expected: matrix of expected cell frequencies, chisq: Pearson X^2 g_squared: likelihood ratio G^2 df: degrees of freedom beta: the parameter estimated sigma_beta: standard error of beta z: z-score for beta delta: transformation of the the parameter into the model formulation

Examples

```
Agresti_simple_diagonals_parameter_quasi_symmetry(vision_data)
```

Agresti_starting_values

Computes staring values for marginal pi.

Description

Computes staring values for marginal pi.

Usage

```
Agresti_starting_values(p)
```

Arguments

p	matrix of observed proportions

Value

vector containing pi

`Agresti_w_diff` *Computes the weighted statistics listed in section 2.3.*

Description

Computes weighted contrast of the two margins. Agresti, A. (1983). Testing marginal homogeneity for ordinal categorical variables. Biometrics, 39(2), 505-510.

Usage

```
Agresti_w_diff(w, n)
```

Arguments

`w`	a vector of weights to be treated as scores
`n`	matrix of observed counts

Value

a list containing diff: the weighted contrast computed using weights w sigma_diff: SE(diff) z_diff: z-score for diff

Examples

```
weights = c(-3.0, -1.0, 1.0, 3.0)
Agresti_w_diff(weights, vision_data)
```

`Agresti_weighted_tau` *Computes weighted tau from Section 2.1. Agresti, A. (1983). Testing marginal homogeneity for ordinal categorical variables. Biometrics, 39(2), 505-510.*

Description

Computes weighted tau from Section 2.1. Agresti, A. (1983). Testing marginal homogeneity for ordinal categorical variables. Biometrics, 39(2), 505-510.

Usage

```
Agresti_weighted_tau(n)
```

Arguments

`n`	matrix of observed counts

Value

a list containing tau: value of tau-d coefficient sigma_tau: SE(tau) z_tau: z-score for tau

`Bhapkar_marginal_homogeneity` *Bhapkar's (1979) test for marginal homogeneity*

Description

Fits the marginal homogeneity model using WLS.

Usage

```
Bhapkar_marginal_homogeneity(n)
```

Arguments

`n`	matrix containing the table to analyze

Details

See: Bhapkar, V. P. (1966). A Note on the Equivalence of Two Test Criteria for Hypotheses in Categorical Data. Journal of the American Statistical Association, 61(313), pp.228-235.

Value

a list containing the chi-square statistic, the df and p-value.

Examples

```
Bhapkar_marginal_homogeneity(vision_data)
```

`Bhapkar_quasi_symmetry` *Bhapkar's 1979 test for quasi-symmetry.*

Description

Fits the quasi-symmetry model using WLS. Bhapkar, V. P. (1979). On tests of marginal symmetry and quasi-symmetry in two and three-dimensional contingency tables. Biometrics 35(2), 417-426.

Usage

```
Bhapkar_quasi_symmetry(n)
```

Arguments

`n`	the matrix to be analyzed

Value

a list containing the chi-square and df.

Examples

```
Bhapkar_quasi_symmetry(vision_data)
```

`Bowker_symmetry` *Computes Bowker's test of symmetry.*

Description

Computes the test of table symmetry in Bowker (1948). Bowker, A. H. (1948). A test for symmetry in contingency tables. Journal of the American Statistical Association 43, 572-574.

Usage

```
Bowker_symmetry(n)
```

Arguments

`n`	the matrix to be tested for symmetry

Value

a list containing the chi-square: Pearson X^2 g_square: likelihood ratio G^2 df: degrees of freedom p-value: p-value for Pearson X^2 expected: fitted values

Examples

```
Bowker_symmetry(vision_data)
```

`budget_actual` *Participation in household budgeting by psychiatric patients. Rows are ratings by patient, columns are ratings by relative. 1 - not at all 2 - doing some 3 - doing regularly*

Description

Participation in household budgeting by psychiatric patients. Rows are ratings by patient, columns are ratings by relative. 1 - not at all 2 - doing some 3 - doing regularly

Usage

```
budget_actual
```

Format

‘budget_actual‘ A matrix with 3 rows and 3 columns

Source

Schuster, C, (2001). Kappa as a parameter of a symmetry model for rater agreement. Journal of Educational and Behavioral Statistics, 26(3), 331-342.

`budget_expected` *Ratings of expected participation in household budgeting by psychiatric patients. Rows are ratings by patient, columns are ratings by relative. 1 - not at all 2 - doing some 3 - doing regularly*

Description

Ratings of expected participation in household budgeting by psychiatric patients. Rows are ratings by patient, columns are ratings by relative. 1 - not at all 2 - doing some 3 - doing regularly

Usage

```
budget_expected
```

Format

‘budget_expected‘ a matrix with 3 rows and 3 columns.

Source

Schuster, C, (2001). Kappa as a parameter of a symmetry model for rater agreement. Journal of Educational and Behavioral Statistics, 26(3), 331-342.

`Clayton_marginal_location` *Fits the tests comparing locations of the margins of a two-way table.*

Description

The measure is based on the weighted cdfs. No "scores" are used, just the weighted (cumulative sums). Clayton, D. G. (1974) Odds ratio statistics for the analysis of ordered categorical data. Biometrika, 61(3), 525-531.

Usage

```
Clayton_marginal_location(wx, wy)
```

Arguments

`wx`	vector containing frequencies for the first margin of the table
`wy`	vector containing frequencies for the second margin of the table

Value

a list of results odds_ratios: odds ratios comparing cumulative frequencies of adjacent categories log_theta_hat: log of estimate of the common odds-ratio theta_hat: estimate of the common odds-ratio log_mh_theta_hat: log of the Mantel-Haenssel type odds-ratio mh_theta_hat: Mantel-Haenszel type odds-ratio var_log_theta_hat = variance of the log of the odds-ratios chisq_theta_hat: chi-square for odds-ratio chisq_mh_theta_hat: chi-square for Mantel-Haenszel odds-ratio df: degrees of freedom for chis-square = 1

Examples

```
Clayton_marginal_location(tonsils[1,], tonsils[2,])
```

Clayton_stratified_marginal_location

Clayton's stratified version of the marginal location comparison.

Description

Compares marginal location conditional on a stratifying variable. Clayton, D. G. (1974) Odds ratio statistics for the analysis of ordered categorical data. Biometrika, 61(3), 525-531.

Usage

```
Clayton_stratified_marginal_location(mx, my)
```

Arguments

`mx`	matrix with
`my`	matrix with

Value

a list of results odds_ratios: odds ratios comparing cumulative frequencies of adjacent categories log_theta_hat: log of estimate of the common odds-ratio theta_hat: estimate of the common odds-ratio log_mh_theta_hat: log of the Mantel-Haenssel type odds-ratio mh_theta_hat: Mantel-Haenszel type odds-ratio var_log_theta_hat = variance of the log of the odds-ratios chisq_theta_hat: chi-square for odds-ratio chisq_mh_theta_hat: chi-square for Mantel-Haenszel odds-ratio df: degrees of freedom for chis-square = 1

See Also

[Clayton_marginal_location()]

`Clayton_summarize` *Computes summary, cumulative proportions up to index provided*

Description

Computes summary, cumulative proportions up to index provided

Usage

```
Clayton_summarize(weights, m)
```

Arguments

`weights`	matrix of counts
`m`	index of summation, weights[1:m]

Value

a list containing: n: the sum of the weights p: matrix of proportion values gamma: cumulative proportions 1:m

`Clayton_summarize_stratified` *Analysis stratified by column variable j.*

Description

Analysis stratified by column variable j.

Usage

```
Clayton_summarize_stratified(weight_matrix, m)
```

Arguments

`weight_matrix`	matrix of cell weights from the table
`m`	the column index to stratify on

Value

a list containing: n: the number of strata p: matrix of proportion values gamma: cumulative proportions

See Also

[Clayton_summarize()]

`Clayton_two_way_association` *Clayton's stratified measure of association*

Description

Quantifies association between two ordinal variables. Clayton, D. G. (1974) Odds ratio statistics for the analysis of oordered categorical data. Biometrika, 61(3), 525-531.

Usage

```
Clayton_two_way_association(f)
```

Arguments

`f`	matrix of frequencies

Value

a list of results log_theta_hat: log odds-ratio measure of association theta_hat: odds-ratio measure of association log_mh_theta_hat: log of Mantel-Haenszel odds-ratio measure of association mh_theta_hat: Mantel-Haenszel odds-ratio measure of association var_log_theta_hat: variance of the log odds-ration measures chisq_theta_hat: chi-square for measure of association chisq_mh_theta_hat: chi-square for Mantel-Haenszel measure of association df: degress of freedom = 1, corr_theta_hat: theta-hat association converted to correlation metric corr_mh_theta_hat: Mantel-Haenszel theta-hat converted to correlation metric

---

`Cliff_as_d_matrix` *Converts two vectors containing scores and integer frequencies (cell counts) into a d-matrix*

Description

Converts two vectors containing scores and integer frequencies (cell counts) into a d-matrix

Usage

```
Cliff_as_d_matrix(scores, cells, nrow = NULL)
```

Arguments

`scores`	vector of scores, typically 1:r
`cells`	vector of integer weights, i.e. cell frequencies
`nrow`	number of score categories in table. Default is NULL. If NULL, takes 1:length(scores)

Value

d-matrix of results

`Cliff_compute_d` *Computes between groups dominance matrix "d".*

Description

Computes between groups dominance matrix "d".

Usage

```
Cliff_compute_d(x, y)
```

Arguments

`x`	first vector of scores
`y`	second vector of scores

Value

N X N dominance matrix

`Cliff_counts_2` *Generates counts from table frequencies for 2 category items*

Description

Generates counts from table frequencies for 2 category items

Usage

```
Cliff_counts_2(mij)
```

Arguments

`mij`	Matrix of counts.

Value

a list containing wm1m1: for -1, -1 wm10: for -1, 0 wm11: for -1, 1 w00: for 0, 0 w01: for 0, 1 w11: for 1, 1

`Cliff_counts_3` *Generates counts from table frequencies for 3 category items*

Description

Generates counts from table frequencies for 3 category items

Usage

```
Cliff_counts_3(mij)
```

Arguments

`mij`	Matrix of counts.

Value

a list containing wm1m1: for -1, -1 wm10: for -1, 0 wm11: for -1, 1 w00: for 0, 0 w01: for 0, 1 w11: for 1, 1

`Cliff_counts_4` *Generates counts from table frequencies for 4 category items*

Description

Generates counts from table frequencies for 4 category items

Usage

```
Cliff_counts_4(mij)
```

Arguments

`mij`	Matrix of counts.

Value

a list containing wm1m1: for -1, -1 wm10: for -1, 0 wm11: for -1, 1 w00: for 0, 0 w01: for 0, 1 w11: for 1, 1

`Cliff_counts_5` *Generates counts from table frequencies for 5 category items*

Description

Generates counts from table frequencies for 5 category items

Usage

```
Cliff_counts_5(mij)
```

Arguments

`mij`	Matrix of counts.

Value

a list containing wm1m1: for -1, -1 wm10: for -1, 0 wm11: for -1, 1 w00: for 0, 0 w01: for 0, 1 w11: for 1, 1

`Cliff_counts_6` *Generates counts from table frequencies for 6 category items*

Description

Generates counts from table frequencies for 6 category items

Usage

```
Cliff_counts_6(mij)
```

Arguments

`mij`	Matrix of counts.

Value

a list containing wm1m1: for -1, -1 wm10: for -1, 0 wm11: for -1, 1 w00: for 0, 0 w01: for 0, 1 w11: for 1, 1

`Cliff_dependent` *Computes Cliff's dependent d-statistics based on a dominance matrix.*

Description

Takes the dominance matrix provided and computes the d-statistics: dw - within-subjects d-statistic db - between-subjects d-statistic db_dw - sum of dw and db, omnibus test of whether one group is higher than the other Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. Psychological Bulletin, 114(3), 494-509. Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mawhaw NJ: Lawerence Erlbaum.

Usage

```
Cliff_dependent(d_matrix)
```

Arguments

`d_matrix`	N x N within-subjects dominance matrix

Value

a list containing dw: within-subjects d-statistic sigma_dw: SE of dw z_dw: z-score for dw db: between-subjects d-statistic sigma_db: SE of db z_db: z-score for db db_dw: sum db + dw, omnibus measure sigma_db_dw: SE of db + dw z_db_dw: z-score of db _ dw cov_db_dw: covariance between db and dw

Examples

```
Cliff_dependent(interference_control_1)
```

`Cliff_dependent_compute_cov` *Computes sum term in covariance db-dw for weighted dominance matrix.*

Description

Computes sum term in covariance db-dw for weighted dominance matrix.

Usage

```
Cliff_dependent_compute_cov(wd)
```

Arguments

`wd`	weighted dominance matrix

Cliff_dependent_compute_cov_from_d

Compute the sum in the covariance of db+dw

Description

Compute the sum in the covariance of db+dw

Usage

```
Cliff_dependent_compute_cov_from_d(d_matrix)
```

Arguments

`d_matrix`	d-matrix of dominances

Value

the sum for the covariance term

Cliff_dependent_compute_from_matrix

Computes Cliff's dependent d-statistics based on a dominance matrix.

Description

Takes the dominance matrix provided and computes the d-statistics: dw - within-subjects d-statistic db - between-subjects d-statistic db_dw - sum of db and dw, omnibus test of whether one group is higher than the other Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. Psychological Bulletin, 114(3), 494-509. Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mawhaw NJ: Lawerence-Erlbaum.

Usage

```
Cliff_dependent_compute_from_matrix(d_matrix)
```

Arguments

`d_matrix`	N x N within-subjects dominance matrix

Value

a list containing dw: within-subjects d-statistic sigma_dw: SE of dw z_dw: z-score for dw db: between-susbjects d-statistic sigma_db: SE of db z_db: z-score for db db_dw: sum db + dw, omnibus measure sigma_db_dw: SE of db + dw z_db_dw: z-score of db _ dw cov_db_dw: covariance between db and dw

Examples

```
Cliff_dependent_compute_from_matrix(interference_control_1)
```

`Cliff_dependent_compute_from_table`
Computes Cliff's dependent d-statistics based on a table of frequency counts.

Description

Takes the r X r table and returns: dw - within-subjects d-statistic db - between-subjects d-statistic db_dw - sum of dw and db, omnibus test of whether one group is higher than the other No intermediate dominance matrix is computed, so this is much faster than Cliff_dependent_compute_from_matrix(). Large number of terms are needed to compute intermediate d_ij_ji. These are contained in separate functions for r <= 6. Results for r [7, 10] are available, but the files are so large that they cause an error if included in the library.

Usage

```
Cliff_dependent_compute_from_table(mij)
```

Arguments

`mij`	an r x r table of paired observations

Details

See: Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. Psychological Bulletin, 114(3), 494-509. Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mawhaw NJ: Lawerence-Erlbaum.

Value

a list containing dw: within-subjects d-statistic sigma_dw: SE of dw z_dw: z-score for dw db: between-susbjects d-statistic sigma_db: SE of db z_db: z-score for db db_dw: sum db + dw, omnibus measure sigma_db_dw: SE of db + dw z_db_dw: z-score of db _ dw cov_db_dw: covariance between db and dw

See Also

[Cliff_dependent_compute_paired_d()]

Examples

```
Cliff_dependent_compute_from_table(movies)
```

`Cliff_dependent_compute_paired_d`
Computes Cliff's dependent d-statistics based on cell frequencies.

Description

Computes d-matrix and then analyzes it. This can be time consuming. Try Cliff_dependent_from_table() instead. The current function is provided mainly for comparison & validation. For an example, compare running this function on vision_data to running Cliff_dependent_from_table(vision_data).

Usage

```
Cliff_dependent_compute_paired_d(cells)
```

Arguments

`cells`	r x r matrix of frequencies

Details

dw - within-subjects d-statistic db - between-subjects d-statistic db_dw - sum of dw and db, omnibus test of whether one group is higher than the other Cliff, N. (1993). Dominance statistics: Ordinal analyses to answer ordinal questions. Psychological Bulletin, 114(3), 494-509. Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mawhaw NJ: Lawerence-Erlbaum.

Value

a list containing dw: within-subjects d-statistic sigma_dw: SE of dw z_dw: z-score for dw db: between-subjects d-statistic sigma_db: SE of db z_db: z-score for db db_dw: sum db + dw, omnibus measure sigma_db_dw: SE of db + dw z_db_dw: z-score of db _ dw cov_db_dw: covariance between db and dw

See Also

[Cliff_dependent_compute_from_table()]

Examples

```
Cliff_dependent_compute_paired_d(movies)
```

Cliff_independent   *Computes the independent groups d-statistic comparing the two vectors provided.*

Description

Computes the independent groups d-statistic comparing the two vectors provided.

Usage

```
Cliff_independent(x, y)
```

Arguments

x	vector of scores for first group
y	vector of scores for second group

Value

list containing d, SE(d) and z(d)

Cliff_independent_from_matrix   *Computes d-statistic from dominance matrix provided.*

Description

Computes d-statistic from dominance matrix provided.

Usage

```
Cliff_independent_from_matrix(d)
```

Arguments

d	N X M dominance matrix

Value

list containing d, SE(d) and z(d)

Cliff_independent_from_table

Computes independent group's d-statistic from the matrix of frequencies provided.

Description

Computes intermediate d-matrix, so can be slow for large N

Usage

```
Cliff_independent_from_table(n)
```

Arguments

n	matrix of counts

Value

list containing d, SE(d) and z(d)

Cliff_independent_weighted

Computes d-statistic based on scores and integer weights(frequencies) for each group.

Description

Computes d-statistic based on scores and integer weights(frequencies) for each group.

Usage

```
Cliff_independent_weighted(x, w_x, y, w_y)
```

Arguments

x	first vector of scores
w_x	weights associated with first vector of scores
y	second vector of scores
w_y	weights associated with second vector of scores

Value

list containing d, SE(d) and z(d)

Cliff_weighted_d_matrix

Computes weighted version of dominance matrix "d"

Description

Arguments are scores and associated weights. Not useful for tables. Use Cliff_compute_d_matrix instead.

Usage

```
Cliff_weighted_d_matrix(x, y, w.x = rep(1, length(x)), w.y = rep(1, length(y)))
```

Arguments

`x`	first vector of scores
`y`	second vector of scores
`w.x`	first vector of weights, to apply to x. Defaults to vector of 1.0
`w.y`	second vector of weights, to apply to y. Defaults to vector of 1.0

Value

an n X m d-matrix, where n is length(x) and m is length(y)

coal_g

Degree of disease measured at two points in time for mine workers.

Description

Based on radiological measurements, the matrix contains the degree of pneumoconiosis in coal workers. 1 = least severe disease and 4 = most severe.

Usage

```
coal_g
```

Format

‘coal_g‘ A matrix with 4 rows and 4 columns.

Source

McCullagh, P. (1977). A logistic model for paired comparisons with ordered categorical data. Biometrika, 64(3), 449-453.

`constant_of_integration` *Computes the constant of integration of a multinomial sample.*

Description

N! / product(n[i]!)

Usage

```
constant_of_integration(n, exclude_diagonal = FALSE)
```

Arguments

`n`	Matrix of observed counts
`exclude_diagonal`	logical. Should the diagonal cells of a square matrix be excluded from the computation. Default is FALSE,

Value

value of constant of integration for observed matrix provided

---

`depression` *Ratings of severity of patient's depression by two therapists.*

Description

1 = slight 2 = moderate 3 = severe

Usage

```
depression
```

Format

'depression' A matrix with 3 rows and 3 columns.

Source

von Eye, A. & Mun, E. Y. (2005, p.41). Analyzing rater agreement: Manifest variable methods. Mahwah, NJ: Lawrence Erlbaum.

`dogs` *Dehydration in dogs data set.*

Description

An interrater agreement data set from Shourki, M. M. (2005, p.80). It is agreement study of two clinicians evaluating whether dogs were dehydrated. The lowest score indicates normal, and the highest score indicates dehydrated (above 10 The "g" in the name indicates that this is taken from mine "G" in the original study.

Usage

```
dogs
```

Format

‘dogs‘ A matrix with 4 rows and 4 columns.

Source

Shoukri, M. M. (2005). The measurement of interobserver agreement. New York: Chapman & Hall.

`dreams` *Severity of disturbing dreams in adolescent boys, measured at two ages..*

Description

Severity of disturbing dreams in adolescent boys, measured at two ages..

Usage

```
dreams
```

Format

‘dreams‘ A matrix with 4 rows and 4 columns.

Source

McCullagh, P. (1980, p.117). Regression models for ordinal data. Journal of the Royal Statistical Society, Series B, 42(2), 109-142.

`dumping` *Occurrence of side effects after gastro-intestinal surgery.*

Description

Columns 1 = None 2 = Slight 3 = Moderate

Usage

```
dumping
```

Format

‘dumping‘ A matrix with 4 rows and 3 columns

Details

Rows Hospital A Hospital B Hospital C Hospital D

Source

Agresti, A. (1984, p. 63). Analysis of ordinal categorical data. Naew York: Wiley.

`esophageal_cancer` *Ratings of number of hot drinks consumed by cases with cancer of the esophagus, compared with control subjects.*

Description

Ratings of number of hot drinks consumed by cases with cancer of the esophagus, compared with control subjects.

Usage

```
esophageal_cancer
```

Format

‘esophageal_cancer‘ A matrix with 4 rows and 4 columns.

Source

Agresti, A. (1984, p. 217). Analysis of ordinal categorical data. New York, Wiley.

`expand` *Converts weighted (x, w) pairs into unweighted data by replicating x[i] w[i] times*

Description

Takes a set of (value, weight) pairs and converts into unweighted vector (w[i]) for each i Weights are assumed to be integers

Usage

```
expand(x, w)
```

Arguments

`x`	Numeric vector of scores.
`w`	Numeric vector of weights. These are assumed to be integers

Value

new unweighted vector of scores

`expit` *Computes the "expit" function – inverse of logit.*

Description

Computes the "expit" function – inverse of logit.

Usage

```
expit(z)
```

Arguments

`z`	Numeric. Real valued argument to expit() function.

Value

exp(z) / (1.0 + exp(z))

`family_income` *Family income for two years from US census.*

Description

Family income for two years from US census.

Usage

```
family_income
```

Format

‘family_income‘ A matrix with 2 rows and 7 columns. Rows are years 1960 and 1970. Columns are income range.

Source

McCullagh, P. (1980, p.114). Regression models for ordinal data. Journal of the Royal Statistical Society, Series B, 42(2), 109-142.

`gender_vision` *Ratings of visual acuity for men and women employed at the Royal Ordinance factories, 1943-1946.*

Description

1 = best visual acuity 4 = worst visual acuity

Usage

```
gender_vision
```

Format

‘gender_vision‘ A matrix with 2 rows for the genders and 4 columns for visual acuity.

Source

McCullagh, P. (1980, p. 119). Regression models for ordinal data. Journal of the Royal Statistical Society, Series B, 42(2), 109-142.

Goodman_constrained_diagonals_parameter_symmetry

Fits the model where some of the delta parameters are constrained to be equal to one another.

Description

Fits the model where some of the delta parameters are constrained to be equal to one another.

Usage

```
Goodman_constrained_diagonals_parameter_symmetry(n, equality)
```

Arguments

n	the matrix of observed counts
equality	logical vector indicating whether corresponding delta the parameter is part of the equality set.

Value

a list containing pooled_chisq: Pearson chi-square for the pooled delta values pooled_df: degrees of freedom for pooled chisq omnibus_chisq: Pearson chi-square for overall model fit, subject to equality constraints omnibus_df; degrees of freedom for omnibus_chisq equality_chisq: Pearson chi-square for test that remaining deltas are all equal equality_df: degrees of freedom for equality_chisq delta_pooled: estimate of pooled delta

Examples

```
equality = c(TRUE, TRUE, FALSE)
Goodman_diagonals_parameter_symmetry(vision_data)
```

Goodman_diagonals_parameter_symmetry

Fit's Goodman's diagonals parameter symmetry model.

Description

Goodman, L. A. (1979). Multiplicative models for square contingency tables with ordered categories. Biometrika, 66(3), 413-316.

Usage

```
Goodman_diagonals_parameter_symmetry(n)
```

Arguments

`n`	the matrix of obsever counts

Value

a list containing individual_chisq: chi-square value for each diagonal individual_df: degrees of freedom for individual_chisq omnibus_chisq: overall chi-square for the model omnibus_df: degrees for freedom for omnibus_chisq equality_chisq: chi-square for test that all delta values are equal equality_df: degrees of freedom from equality_chisq delta: the vector of estimated delta values (without any equality constraints)

Examples

```
Goodman_diagonals_parameter_symmetry(vision_data)
```

`Goodman_fixed_parameter` *Fits the model with given parameters fixed to specific values.*

Description

The model has simple closed form solutions when fitting either the unconstrained version of the version that species equality of delta parameters. However, I could not see how to adapt that to the case where specific parameters were constrained to have a specific value. This routine is to fit that model. It will also fit the unconstrained model, but Goodman gives the estimator for that case.

Usage

```
Goodman_fixed_parameter(
  n,
  delta,
  fixed,
  convergence = 1e-04,
  max_iter = 50,
  verbose = FALSE
)
```

Arguments

`n`	the r X r matrix of observed counts
`delta`	the vector of asymmetry r - 1 parameters
`fixed`	r - 1 logical vector that specifies whether a delta parameter is fixed (TRUE) or allowed to be estimated (FALSE).
`convergence`	maximum change in a parameter across iterations. Default is 1.0e-4
`max_iter`	maximum number of iterations, Default is 50.
`verbose`	should progress information be printed to the console. Default is FALSE, do not print.

Value

list containing phi, delta, max_change largest change in parameter for last the iteration, chisq: Pearson chi-square g_squared: likelihood ratio G^2 df: degrees of freedom

See Also

[Goodman_diagonals_parameter_symmetry()]

[Goodman_ml()]

Examples

```
fixed <- c(FALSE, TRUE, FALSE)
delta <- c(1.0, 1.0, 1.0)
phi <- matrix(0.0, nrow=4, ncol=4)
diag(phi) = rep(1.0, 4)
Goodman_fixed_parameter(vision_data, delta, fixed)
```

Goodman_ml *Performs ML estimation of the model.*

Description

The model has simple closed form solutions when fitting either the unconstrained version of the version that species equality of delta parameters. However, I could not see how to adapt that to the case where specific parameters were constrained to have a specific value. This routine is to fit that model. It will also fit the unconstrained model, but Goodman gives the estimator for that case.

Usage

```
Goodman_ml(n, phi, delta, fixed)
```

Arguments

`n`	the r X r matrix of observed counts
`phi`	the symmetric matrix parameter
`delta`	the vector of asymmetry r - 1 parameters
`fixed`	r - 1 logical vector that specifies whether a delta parameter is fixed (TRUE) or allowed to be estimated (FALSE).

Value

list containing new estimates of phi amd delta

See Also

[Goodman_diagonals_parameter_symmetry()]

Examples

```
fixed <- c(FALSE, TRUE, FALSE)
delta <- c(1.0, 1.0, 1.0)
phi <- matrix(0.0, nrow=4, ncol=4)
for (i in 1:4) {
  phi[i, i] = 1.0
}
Goodman_ml(vision_data, phi, delta, fixed)
```

`Goodman_model_i` *Fits Goodman's (1979) Model I*

Description

Fits Goodman's (1979) Model I

Usage

```
Goodman_model_i(
  n,
  row_effects = TRUE,
  column_effects = TRUE,
  max_iter = 25,
  verbose = FALSE,
  exclude_diagonal = FALSE
)
```

Arguments

`n`	matrix of observed counts
`row_effects`	should row effects be included in the model? Default is TRUE
`column_effects`	should column effects be included in the model? Default is TRUE
`max_iter`	maximum number of iterations. Default is 10
`verbose`	logical. Should cycle-by-cycle output be printed? Default is no
`exclude_diagonal`	logical. For square tables, should the cells on the diagonal be excluded? Default is FALSE, include all cells

Value

a list containing alpha: row effects beta: column effects gamma: row location weights delta: column location weights log_likelihood: log(likelihood) g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom

Goodman_model_i_star *Fits Goodman's (1979) Model I**

Description

Fits Goodman's (1979) Model I*

Usage

```
Goodman_model_i_star(
  n,
  max_iter = 25,
  verbose = FALSE,
  exclude_diagonal = FALSE
)
```

Arguments

n	matrix of observed counts
max_iter	maximum number of iterations
verbose	should cycle-by-cycle information be printed out? Default is FALSE, do not print
exclude_diagonal	should the cells along the main diagonal be excluded? Default is FALSE, include all cells

Value

a list containing alpha: vector of row parameters beta: vector of column parameters theta: vector of common row/column estimates log_likelihood: log(likelihood) at completion g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom

Goodman_model_ii *Fits Goodman's (1979) Model II*

Description

Fits Goodman's (1979) Model II

Usage

```
Goodman_model_ii(
  n,
  rho = 1:nrow(n) - (nrow(n) + 1)/2,
  sigma = 1:ncol(n) - (ncol(n) + 1)/2,
  update_rows = TRUE,
  update_columns = TRUE,
  max_iter = 25,
  verbose = FALSE,
  exclude_diagonal = FALSE
)
```

Arguments

n	matrix of observed counts
rho	values of row locations. Default is 1:nrow(n) - (nrow(n) + 1) / 2
sigma	values of column locations. Default is 1:ncol(n) - (ncol(n) + 1) / 2
update_rows	should values of row locations be updated? Default is TRUE, update
update_columns	should value of column locations be updated? Default is TRUE, update
max_iter	maximum number of iterations to perform. Default is 10
verbose	should cycle-by-cycle output be produced? Default is FALSE
exclude_diagonal	logical. Should the diagonal be excluded from the computation. Default is FALSE.

Value

a list containing alpha: row effects beta: column effects rho: centered row locations mu: row locations sigma: centered column locations nu: column locations log_likelihood: log(likelihood) g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom

---

Goodman_model_ii_star *Fits Goodman's (1979) model II*, where row and column effects are equal.*

Description

Fits Goodman's (1979) model II*, where row and column effects are equal.

Usage

```
Goodman_model_ii_star(
  n,
  exclude_diagonal = FALSE,
  max_iter = 25,
  verbose = FALSE
)
```

Arguments

n	matrix of observed counts
exclude_diagonal	should the cells of the main diagonal be excluded? Default is FALSE, include all cells
max_iter	maximum number of iterations
verbose	should cycle-by-cycle information be printed out? Default is FALSE, do not print

Value

a list containing alpha: vector of alpha (row) parameters beta: vector of beta (column) parameters phi: vector of common row/column effects log_likelihood: value of the log(likelihood) function at completion g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom

---

Goodman_null_association

Fits Goodman's L. A. (1979) Simple Models for the Analysis of Association in Cross-Classifications Having Ordered Categories

---

Description

null association model

Usage

```
Goodman_null_association(
  n,
  max_iter = 25,
  verbose = FALSE,
  exclude_diagonal = FALSE
)
```

Arguments

n	matrix of observed counts
max_iter	maximum number of iterations. Default is 10
verbose	should cycle-by-cycle info be printed? Default is FALSE
exclude_diagonal	logical, Should the diagonal be excluded from the computations. Default is FALSE

Value

a list containing alpha: row effects beta: column effects log_likelihood: log(likelihood) g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom

`Goodman_pi` *Computes the model-based probability for cell i, j*

Description

Computes the model-based probability for cell i, j

Usage

```
Goodman_pi(phi, delta, i, j)
```

Arguments

`phi`	symmetry matrix
`delta`	vector of asymmetry parameters
`i`	row index
`j`	column index

Value

pi for that cell

`Goodman_pi_matrix` *Computes the full matrix of model-based cell probabilities.*

Description

Computes the full matrix of model-based cell probabilities.

Usage

```
Goodman_pi_matrix(phi, delta)
```

Arguments

`phi`	the symmetric matrix
`delta`	the vector of asymmetry parameters

Value

matrix of model-based probabilities

Goodman_symmetric_association_model

Fits the symmetric association model from Goodman (1979). Note the model is a reparameterized version of the quasi-symmetry model, so the quasi-symmetry model has the same fit indices.

Description

Fits the symmetric association model from Goodman (1979). Note the model is a reparameterized version of the quasi-symmetry model, so the quasi-symmetry model has the same fit indices.

Usage

```
Goodman_symmetric_association_model(n)
```

Arguments

`n`	matrix of observed counts

Value

a list containing x: design matrix used for the glm() regression beta: parameter estimates se: standard errors of beta g_squared: G^2 measure of fit chisq: X^2 measure of fit df: degrees of freedom expected: model-based expected cell counts

Goodman_uniform_association

Fits Goodman's (1979) uniform association model

Description

Fits Goodman's (1979) uniform association model

Usage

```
Goodman_uniform_association(
  n,
  max_iter = 25,
  verbose = FALSE,
  exclude_diagonal = FALSE
)
```

Arguments

`n`	matrix of observed counts
`max_iter`	maximum number of iterations. Default is 10.
`verbose`	should cycle-by-cycle info be printed out? Default is FALSE
`exclude_diagonal`	logical. Should the cells of the main diagonal be excluded from the computations? Default is FALSE, include all cells.

Value

a list containing alpha: row effects beta: column effects theta: uniform association parameter log_likelihood: log(likelihood) g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom

---

`handle_max_i_i` *Case where j == r, i == k == k2*

Description

Case where j == r, i == k == k2

Usage

```
handle_max_i_i(i, marginal_pi, kappa, v)
```

Arguments

`i`	index into marginal_pi
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

second-order derivative

handle_max_i_k *Case where j == r, i != k, i == k2*

Description

Case where j == r, i != k, i == k2

Usage

```
handle_max_i_k(i, k, marginal_pi, kappa, v)
```

Arguments

`i`	index into pi
`k`	index into v (other is i)
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

second-order derivative

handle_max_k_k2 *Case where j == r, i != k && i != k2*

Description

Case where j == r, i != k && i != k2

Usage

```
handle_max_k_k2(i, k, k2, marginal_pi, kappa, v)
```

Arguments

`i`	index into pi
`k`	first index into marginal_pi
`k2`	second index into marginal_pi
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

second-order derivative

handle_one_maximum *Case where pi[i, r] with k and k2*

Description

Case where pi[i, r] with k and k2

Usage

```
handle_one_maximum(i, j, k, k2, marginal_pi, kappa, v)
```

Arguments

i	first index of pi
j	second index of pi
k	first index into marginal_pi
k2	second index into marginal_pi
marginal_pi	expected proportions for each category
kappa	current estimate of kappa coefficient
v	symmetry matrix

Value

second order derivative

handle_tied_below_maximum *Case where i == j, i < r, j < r*

Description

Case where i == j, i < r, j < r

Usage

```
handle_tied_below_maximum(j, k, k2, marginal_pi, kappa, v)
```

Arguments

j	index of pi
k	first index into marginal_pi
k2	second index into marginal_pi
marginal_pi	expected proportions for each of the categories
kappa	current estimate of kappa coefficient
v	symmetry matrix

Value

derivative

---

handle_tied_maximum *Case where pi[r, r] with k and k2*

Description

Case where pi[r, r] with k and k2

Usage

```
handle_tied_maximum(k, k2, marginal_pi, kappa, v)
```

Arguments

`k`	first index into marginal_pi
`k2`	second index into marginal_pi
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

second order derivative

---

handle_untied_below_maximum *Case where i != j, i < r && j < r*

Description

Case where i != j, i < r && j < r

Usage

```
handle_untied_below_maximum(i, j, k, k2, marginal_pi, kappa, v)
```

Arguments

i	first index of pi
j	second index of pi
k	first index of marginal_pi
k2	second index of marginal_pi
marginal_pi	expected proportions of each of the categories
kappa	current value of kappa coefficient
v	symmetry matrix

---

homicide_black_black *Data about charges of homicide in the state of Florida.*

Description

Counts of cases charged with homicide. The rows and columns indicate whether there was an additional charge of a felony occurring in addition to the homicide. The data is actually 3-dimensional. It is stored as 4 related matrices, each with the leading word "homicide_" The rest of the name gives the race of the defendant and the race of the victim, separated by an underscore

Usage

```
homicide_black_black
```

Format

‘homicide_black_black‘ Each is a matrix with 3 rows and 3 columns. Rows are classification by police and columns are classification by the court/prosecutor. 1 = No felony 2 = Possible felony 2 = Felony

Source

Agresti, A. (1984, p. 211). Analysis of ordinal categorical data. New York: Wiley.

homicide_black_white *Data about charges of homicide in the state of Florida.*

Description

Counts of cases charged with homicide. The rows and columns indicate whether there was an additional charge of a felony occurring in addition to the homicide. The data is actually 3-dimensional. It is stored as 4 related matrices, each with the leading word "homicide_" The rest of the name gives the race of the defendant and the race of the victim, separated by an underscore.

Usage

```
homicide_black_white
```

Format

'homicide_black_white' Each is a matrix with 3 rows and 3 columns. Rows are classification by police and columns are classification by the court/prosecutor. 1 = No felony 2 = Possible felony 2 = Felony

Source

Agresti, A. (1984, p. 211). Analysis of ordinal categorical data. New York: Wiley.

homicide_white_black *Data about charges of homicide in the state of Florida.*

Description

Counts of cases charged with homicide. The rows and columns indicate whether there was an additional charge of a felony occurring in addition to the homicide. The data is actually 3-dimensional. It is stored as 4 related matrices, each with the leading word "homicide_" The rest of the name gives the race of the defendant and the race of the victim, separated by an underscore

Usage

```
homicide_white_black
```

Format

'homicide_white_black' Each is a matrix with 3 rows and 3 columns. Rows are classification by police and columns are classification by the court/prosecutor. 1 = No felony 2 = Possible felony 2 = Felony

Source

Agresti, A. (1984, p. 211). Analysis of ordinal categorical data. New York: Wiley.

`homicide_white_white` *Data about charges of homicide in the state of Florida.*

Description

Counts of cases charged with homicide. The rows and columns indicate whether there was an additional charge of a felony occurring in addition to the homicide. The data is actually 3-dimensional. It is stored as 4 related matrices, each with the leading word "homicide_" The rest of the name gives the race of the defendant and the race of the victim, separated by an underscore

Usage

```
homicide_white_white
```

Format

‘homicide_white_white‘ Each is a matrix with 3 rows and 3 columns. Rows are classification by police and columns are classification by the court/prosecutor. 1 = No felony 2 = Possible felony 2 = Felony

Source

Agresti, A. (1984, p. 211). Analysis of ordinal categorical data. New York: Wiley.

`hypothalamus_1` *Measures of men’s hypothalamus taken from cadavers. First data set.*

Description

Measures of men’s hypothalamus taken from cadavers. First data set.

Usage

```
hypothalamus_1
```

Format

‘hypothalamus_1‘ Each set is a dominance matrix (see e.g., Cliff 1996).

Source

Cliff, N. (1996), Ordinal methods for behavioral data analysis. Mahwah NJ: Lawrence Erlbaum.

hypothalamus_2 *Measures of men's hypothalamus taken from cadavers. Second data set.*

Description

Measures of men's hypothalamus taken from cadavers. Second data set.

Usage

```
hypothalamus_2
```

Format

‘hypothalamus_2‘ Each set is a dominance matrix (see e.g., Cliff 1996).

Source

Cliff, N. (1996), Ordinal methods for behavioral data analysis. Mahwah NJ: Lawrence Erlbaum.

interference_12 *Measures of interference in memory recall study.*

Description

Measures are within subjects, comparing a control condition to two conditions with interference. Interference condition 1 v. interference condition 2

Usage

```
interference_12
```

Format

‘interference_control_1‘, ‘interference_control_2‘, ‘interference_12‘ Within-persons dominance matrices.

Source

Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mahwah NJ: Lawrence Erlba interference_control_1

Measures of interference in memory recall study.

Description

Measures are within subjects, comparing a control condition to two conditions with interference. Control v. interference condition 1

Usage

```
interference_control_1
```

Format

‘interference_control_1‘, ‘interference_control_2‘, ‘interference_12‘ Within-persons dominance matrices.

Source

Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mahwah NJ: Lawrence Erlbaum.

interference_control_2

Measures of interference in memory recall study.

Description

Measures are within subjects, comparing a control condition to two conditions with interference. Control v. interference condition 2

Usage

```
interference_control_2
```

Format

‘interference_control_1‘, ‘interference_control_2‘, ‘interference_12‘ Within-persons dominance matrices.

Source

Cliff, N. (1996). Ordinal methods for behavioral data analysis. Mahwah NJ: Lawrence Erlba

`Ireland_marginal_homogeneity` *Fits marginal homogeneity model*

Description

Fits the marginal homogeneity model according to the minimum discriminant information. Ireland, C. T., Ku, H. H., & Kullback, S. (1969). Symmetry and marginal homogeneity of an r × r contingency table. Journal of the American Statistical Association, 64(328), 1323-1341.

Usage

```
Ireland_marginal_homogeneity(
  n,
  truncated = FALSE,
  max_iter = 15,
  verbose = FALSE
)
```

Arguments

n	matrix of observed counts
truncated	should the diagonal be excluded. Default is FALSE, include the diagonal.
max_iter	maximum number of iterations to perform
verbose	should cycle-by-cycle information be printed out. Default is FALSE.

Value

a list containing mdis: value of the minimum discriminant information statistic (appox chi-squared) df: dgrees of freedom x_star: matrix of model-based counts p_star: matrix of model-based p-values

Examples

```
Ireland_marginal_homogeneity(vision_data)
```

`Ireland_mdis` *Computes the MDIS between the two matrices provided.*

Description

Computes the MDIS between the two matrices provided.

Usage

```
Ireland_mdis(n, x_star, truncated = FALSE)
```

Arguments

`n`	first matrix (usually observed counts)
`x_star`	second matrix (usually model-based)
`truncated`	should the diagonal be ignored. Default is FALSE, include the diagonal elements.

Value

value of the MDIS criterion

`Ireland_normalize_for_truncation`
Renormalize counts to account for truncation of diagonal

Description

Renormalize counts to account for truncation of diagonal

Usage

```
Ireland_normalize_for_truncation(n)
```

Arguments

`n`	matrix of observed counts

Value

matrix n with diagonal set to 0.0

`Ireland_quasi_symmetry`
Fit for quasi-symmetry model. Obtained by subtraction, so no model-based probabilities.

Description

Fit for quasi-symmetry model. Obtained by subtraction, so no model-based probabilities.

Usage

```
Ireland_quasi_symmetry(n, truncated = FALSE)
```

Arguments

`n`	matrix of observed counts
`truncated`	should the diagonal be excluded, Default is FALSE, include the diagonal.

Value

a list with mdis = MDIS value and df = degrees of freedom for quasi-symmetry model

See Also

[Ireland_quasi_symmetry_model()]

Examples

```
Ireland_quasi_symmetry(vision_data)
```

Ireland_quasi_symmetry_model

Fitss the quasi-symmetry model.

Description

Fits the model according to the MDIS criterion.

Usage

```
Ireland_quasi_symmetry_model(
  n,
  truncated = FALSE,
  max_iter = 5,
  verbose = FALSE
)
```

Arguments

`n`	matrix of observed counts
`truncated`	should the diagonal be excluded. Default is FALSE, include diagonal cells.
`max_iter`	maximum number of iterations in minimizing the criterion. Default is 4
`verbose`	logical variable, should cycle-by-cycle info be printed. Defaullt is FALSE.

Value

a list containing mdis: value of the MDIS at termination df: degrees of freedom x_star: matrix of model-reproduced counts p_star: matrix of model-reproduced p-values

See Also

[Ireland_quasi_symmetry()]

Examples

```
Ireland_quasi_symmetry_model(vision_data)
```

Ireland_symmetry   *Fits symmetry model.*

Description

Ireland, C. T., Ku, H. H., & Kullback, S. (1969). Symmetry and marginal homogeneity of an r × r contingency table. Journal of the American Statistical Association, 64(328), 1323-1341.

Usage

```
Ireland_symmetry(n, truncated = FALSE)
```

Arguments

`n`	matrix of observed counts
`truncated`	should the diagonal be excluded. Default is FALSE, include the diagonal.

Value

a list containing mdis: value of the minimum discriminant information statistic (appox chi-squared) df: dgrees of freedom x_star: matrix of model-based counts p_star: matrix of model-based p-values

Examples

```
Ireland_symmetry(vision_data)
```

---

is_invertible   *Tests whether a square matrix is invertible (non singular)*

Description

from stackoverflow: https://stackoverflow.com/questions/24961983/how-to-check-if-a-matrix-has-an-inverse-in-the-r-language

Usage

```
is_invertible(X)
```

Arguments

`X`	Matrix to be tested. It is assumed X is square

Value

logical: TRUE if inversion succeeds, FALSE otherwise

`is_missing_or_infinite` *Determines if its argument is not a valid number.*

Description

Determines if its argument is not a valid number.

Usage

```
is_missing_or_infinite(x)
```

Arguments

`x`	Numeric. Number of be evaluated

Value

TRUE if is.na(), is.nan(), or is.infinite() returns TRUE. FALSE otherwise.

`kappa` *Computes Cohen's 1960 kappa coefficient*

Description

Computes Cohen's 1960 kappa coefficient

Usage

```
kappa(n)
```

Arguments

`n`	matrix of observed counts

Value

kappa coefficient

likelihood_ratio_chisq *Computes the likelihood ratio G^2 measure of fit.*

Description

Computes the likelihood ratio G^2 measure of fit.

Usage

```
likelihood_ratio_chisq(n, pi, exclude_diagonal = FALSE)
```

Arguments

n	Matrix of observed counts
pi	Matrix of same dimensions as n. Model-based matrix of predicted proportions
exclude_diagonal	logical. Should the diagonal cells of a square matrix be excluded from the computation. Default is FALSE. The effect of setting it to TRUE for non-square matrices may be unintuitive and should he avoided.

Value

G^2

loadRData *Function to load a data set written out using save().*

Description

The first (should be the only) element read from the RData file is returned From: https://stackoverflow.com/questions/5577221 can-i-load-an-object-into-a-variable-name-that-i-specify-from-an-r-data-file

Usage

```
loadRData(file_name)
```

Arguments

file_name	Character. Name of the file containing the RData

Details

usage x <- loadRData(file_name="")

Value

the first object from the restored RData

log_likelihood — *Computes the multinomial log(likelihood).*

Description

Computes the multinomial log(likelihood).

Usage

```
log_likelihood(n, pi, exclude_diagonal = FALSE)
```

Arguments

n	Matrix of observed counts
pi	Matrix of same dimensions as n. Model-based matrix of predicted proportions
exclude_diagonal	logical. Should diagonal cells of square matrix be excluded from the computation? Default is FALSE. The effect of setting it to TRUE for non-square matrices may be unintuitive and should he avoided.

Value

log(likelihood)

log_linear_add_all_diagonals — *Adds indicator variables for the diagonal cells in table n.*

Description

Adds indicator variables for the diagonal cells in table n.

Usage

```
log_linear_add_all_diagonals(n, x)
```

Arguments

n	the matrix of observed counts
x	the design matrix to be augmented

Value

new design matrix with nrow(n) columns added. The columns are all 0 unless the row corresponds to a diagonal cell in n, in which case the entry is 1

Examples

```
x <- log_linear_main_effect_design(vision_data)
x_prime <- log_linear_add_all_diagonals(vision_data, x)
```

---

log_linear_append_column *Appends a column to an existing design matrix.*

Description

Takes the design matrix provided and appends the new column

Usage

```
log_linear_append_column(x, x_new, position = ncol(x) + 1)
```

Arguments

x	the original design matrix
x_new	the column to be appended
position	column index within the new matrix for the new column. Defaults to last position = appending the column

Value

the new design matrix

Examples

```
x <- log_linear_main_effect_design(vision_data)
new_column <- c(1, 0, 0, 0,
                0, 1, 0, 0,
                0, 0, 1, 0,
                0, 0, 0, 1)
x_prime <- log_linear_append_column(x, new_column)
```

log_linear_create_coefficient_names

Creates missing column names

Description

Creates missing column names

Usage

```
log_linear_create_coefficient_names(x, n, effect_names = NULL)
```

Arguments

`x`	the design matrix being modified
`n`	the matrix of observed counts
`effect_names`	user specified names to be applied to effects after the intercept and main effects. Default is NULL

Value

vector of names to apply to x

log_linear_create_linear_by_linear

Creates a vector containing the linear-by-linear vector.

Description

Uses the ordinal ranks (1, 2, ..., nrow(n)) as data.

Usage

```
log_linear_create_linear_by_linear(n, centered = FALSE)
```

Arguments

`n`	the matrix of observed cell counts
`centered`	should the variables be centered before the product is computed

Value

a vector containing the new variable

Examples

```
linear <- log_linear_create_linear_by_linear(vision_data)
x <- log_linear_equal_weight_agreement_design(vision_data)
x_prime <- log_linear_append_column(x, linear)
```

---

log_Linear_create_log_n

Computes the logs of the cell frequencies.

Description

In the case of an observed 0, epsilon is inserted into the cell before the log is taken.

Usage

```
log_Linear_create_log_n(n, epsilon = 1e-06, all_cells = FALSE)
```

Arguments

n	matrix of cell counts
epsilon	amount to be inserted into cell with observed 0.
all_cells	add epsilon to all cells or just those with 0 observed frequencies

Value

a list containing: log_n – a vector of log frequencies and dat – modified version of the cell counts data

---

log_linear_equal_weight_agreement_design

Creates design matrix for model with main effects and a single agreement parameter delta.

Description

The model has main effects for rows and for columns, plus an additional parameter for the agreement (diagonal) cells.

Usage

```
log_linear_equal_weight_agreement_design(n, n_raters = 2)
```

Arguments

`n`	the matrix of cell counts
`n_raters`	number of raters. Currently only 2 (the default) are supported. This is an extension point for future work.

Value

design matrix for the model

Examples

```
x <- log_linear_equal_weight_agreement_design(vision_data)
```

`log_linear_fit` *Fits a log-linear model to the data provided, using the design matrix provided. Names for the effects will be "rows1", "cols1" etc. If there are remaining entries, they can be specified as the "effect_names" character vector. This function is a wrapper around a call to glm() that handles some of the details of the call and packages the output in a more convenient form.*

Description

Fits a log-linear model to the data provided, using the design matrix provided. Names for the effects will be "rows1", "cols1" etc. If there are remaining entries, they can be specified as the "effect_names" character vector. This function is a wrapper around a call to glm() that handles some of the details of the call and packages the output in a more convenient form.

Usage

```
log_linear_fit(n, x, effect_names = NULL)
```

Arguments

`n`	matrix of observed counts to be fit
`x`	design matrix for predictor variables
`effect_names`	character vector of additional names to apply to the columns of x The default is NULL, in which case the columns will be labeled "model1" etc.

Value

a list containing x: the design matrix beta: the regression parameters se: the vector of standard errors g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

log_linear_main_effect_design

Design matrix for baseline independence model with main effects for rows and columns.

Description

It is intended as a straw-man model as it assumes no agreement beyond chance.

Usage

```
log_linear_main_effect_design(n, n_raters = 2)
```

Arguments

n	the matrix of cell counts
n_raters	number of raters. Currently only 2 (the default) are supported. This is an extension point for future work.

Value

the design matrix for the model

Examples

```
x <- log_linear_main_effect_design(vision_data)
```

log_linear_matrix_to_vector

Converts a matrix of data into a vector suitable for use in analysis with the design matrices created. Unlike simply calling vector() on the matrix the resulting vector is organized by rows, then columns. This order corresponds to the order in the design matrix.

Description

Converts a matrix of data into a vector suitable for use in analysis with the design matrices created. Unlike simply calling vector() on the matrix the resulting vector is organized by rows, then columns. This order corresponds to the order in the design matrix.

Usage

```
log_linear_matrix_to_vector(dat)
```

Arguments

`dat`	the matrix to be converted a vector

Value

a vector suitable to use as dependent variable, e.g. in a call to glm()

`log_linear_quasi_symmetry_model_design` *Creates the design matrix for a quasi-symmetry design*

Description

Creates the design matrix for a quasi-symmetry design

Usage

```
log_linear_quasi_symmetry_model_design(n)
```

Arguments

`n`	matrix of observed counts

Value

design matrix for quasi-symmetry design

`log_linear_remove_column` *Removes a column from an existing design matrix.*

Description

Takes the design matrix provided and removes the column in the position specified

Usage

```
log_linear_remove_column(x, position = ncol(x))
```

Arguments

`x`	the original design matrix
`position`	column index within the new matrix for the new column. Defaults to last position

Value

the new design matrix

Examples

```
x <- log_linear_main_effect_design(vision_data)
linear <- log_linear_create_linear_by_linear(vision_data)
x_prime <- log_linear_append_column(x, linear)
x_again <- log_linear_remove_column(x_prime, ncol(x_prime))
```

---

log_linear_symmetry_design — *Creates design matrix for symmetry model.*

Description

Creates design matrix for symmetry model.

Usage

```
log_linear_symmetry_design(n)
```

Arguments

n	matrix of observed counts

Value

design matrix for the model

---

logit — *Computes the log-odds (logit) for the value provided*

Description

Computes the log-odds (logit) for the value provided

Usage

```
logit(p)
```

Arguments

p	Numeric. Assumed to lie in interval(0, 1)

Value

log(p / (1.0 - p))

McCullagh_compute_c_plus

Computes sums c+ used in maximizing the log(likelihod)

Description

Computes sums c+ used in maximizing the log(likelihod)

Usage

```
McCullagh_compute_c_plus(phi, alpha)
```

Arguments

phi	matrix of symmetry parameters
alpha	vector of asymmetry parameters

Value

list of c_i_plus and c_plus_i

McCullagh_compute_condition

Compute the linear constraint on psi elements for identifiablity.

Description

Compute the linear constraint on psi elements for identifiablity.

Usage

```
McCullagh_compute_condition(psi)
```

Arguments

psi	symmetry matrix

Value

value of the constraint

McCullagh_compute_cumulative_sums

Computes cumulative sums for rows,

Description

Computes cumulative sums for rows,

Usage

```
McCullagh_compute_cumulative_sums(n)
```

Arguments

n	matrix of observed counts

Value

R where R[i,] contains cumulative sum of n[i,]

McCullagh_compute_cumulatives

Computes the model-based cumulative probability matrices pij and qij

Description

Computes the model-based cumulative probability matrices pij and qij

Usage

```
McCullagh_compute_cumulatives(psi, delta, alpha, c = 1)
```

Arguments

psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

list containing matrices pij and qij

`McCullagh_compute_df` *Computes the degrees of freedom for the model*

Description

Computes the degrees of freedom for the model

Usage

```
McCullagh_compute_df(M, generalized = FALSE)
```

Arguments

`M`	the size of the M X M observed matrix
`generalized`	is the generalized model being fit? Default is FALSE, regular model

---

`McCullagh_compute_gamma` *Computes gamma from x and beta*

Description

Computes gamma from x and beta

Usage

```
McCullagh_compute_gamma(x, beta, s, c)
```

Arguments

`x`	predictor variables
`beta`	vector of regression coefficients
`s`	number of rows in the table
`c`	number of score levels in table

Value

vector of model-based gamma coefficients

McCullagh_compute_gamma_from_phi

Computes value of gamma from phi. Inverse of usual computation.

Description

Computes value of gamma from phi. Inverse of usual computation.

Usage

```
McCullagh_compute_gamma_from_phi(phi, j, gamma)
```

Arguments

phi	value to compute from
j	index to use in computation
gamma	vector of gamma values (model-based cumulative logits)

Value

gamma[j] given phi and gamma[j + 1]

McCullagh_compute_gamma_plus_1_from_phi

Computes value of gamma[j + 1] from phi.

Description

Computes value of gamma[j + 1] from phi.

Usage

```
McCullagh_compute_gamma_plus_1_from_phi(phi, j, gamma)
```

Arguments

phi	value used in computation
j	index to use in computation
gamma	vector of gamma values (model-based cumulative logits)

Value

gamma[j + 1] given phi and gamma[j]

McCullagh_compute_generalized_cumulatives

Coompute the model-based cumulative probabilities pij and qij.

Description

Coompute the model-based cumulative probabilities pij and qij.

Usage

```
McCullagh_compute_generalized_cumulatives(psi, delta_vec, alpha, c = 1)
```

Arguments

`psi`	symmetry matrix
`delta_vec`	vector of asymmetry parameters
`alpha`	vector of asymmetry parameters
`c`	normalizing constant so pis sum to 1. Defaults to 1.0

Value

matrices of model-based cumulative probabilities pij and qij

McCullagh_compute_generalized_pi

Cpompute matrix pi under generalized model.

Description

Cpompute matrix pi under generalized model.

Usage

```
McCullagh_compute_generalized_pi(psi, delta_vec, alpha, c = 1)
```

Arguments

`psi`	the matrix of symmetry parameters
`delta_vec`	the vector asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

the matrix pi

McCullagh_compute_lambda
Computes lambda, log of cumulative odds.

Description

Computes lambda, log of cumulative odds.

Usage

```
McCullagh_compute_lambda(n, use_half = TRUE)
```

Arguments

n	matrix of observed counts
use_half	logical whether of not to add half to the cell count before taking the logit. Default value is TRUE.

McCullagh_compute_log_l
Computes the log(likelihood) for the general nonlinear model.

Description

Computes the log(likelihood) for the general nonlinear model.

Usage

```
McCullagh_compute_log_l(n, phi)
```

Arguments

n	matrix of observed counts
phi	vector of model-based parameters

Value

log(likelihood)

McCullagh_compute_Nij *Compute the observed sums Nij*

Description

Compute the observed sums Nij

Usage

```
McCullagh_compute_Nij(n)
```

Arguments

n	the matrix of observed counts

Value

a list containing Pij and Qij

McCullagh_compute_omega *Compute the value of the Lagrange multiplier for the constraint on psi.*

Description

Compute the value of the Lagrange multiplier for the constraint on psi.

Usage

```
McCullagh_compute_omega(n, pi)
```

Arguments

n	matrix of observed counts
pi	matrix of model-based probabilities pi.

Value

the value of the Lagrange multiplier.

McCullagh_compute_phi *Computes phi based on gamma*

Description

Computes phi based on gamma

Usage

```
McCullagh_compute_phi(gamma, j)
```

Arguments

gamma	vector of gamma parameters
j	index of phi to compute

Value

phi[j]

McCullagh_compute_phi_matrix *Compute matrix of model-based logits*

Description

Compute matrix of model-based logits

Usage

```
McCullagh_compute_phi_matrix(gamma)
```

Arguments

gamma	matrix of model-based cumulative odds

Value

matrix of model-based logits

McCullagh_compute_pi — *Compute the regular (non-cumulative) model-based pi values*

Description

Compute the regular (non-cumulative) model-based pi values

Usage

```
McCullagh_compute_pi(psi, delta, alpha, c)
```

Arguments

`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

the matrix pi

McCullagh_compute_pi_from_beta — *Computes matrix of p-values pi based on x and current value of beta.*

Description

Computes matrix of p-values pi based on x and current value of beta.

Usage

```
McCullagh_compute_pi_from_beta(n, x, beta)
```

Arguments

`n`	matrix of observed counts
`x`	design matrix
`beta`	current values of location model regression parameters

Value

matrix of model-based pi values

`McCullagh_compute_pi_from_gamma`
Compute the cell probabilities pi from gamma.

Description

Compute the cell probabilities pi from gamma.

Usage

```
McCullagh_compute_pi_from_gamma(gamma)
```

Arguments

`gamma`	matrix of gamma values

Value

c X c matrix of p-values pi

`McCullagh_compute_regression_weights`
*Computes regression weights w; R_dot_j * (N - R_dot_j[j]) * (n_do_j[j] a= na_dot_j[j+ 1])*

Description

Computes regression weights w; R_dot_j * (N - R_dot_j[j]) * (n_do_j[j] a= na_dot_j[j+ 1])

Usage

```
McCullagh_compute_regression_weights(n)
```

Arguments

`n`	matrix of observed counts

Value

list of w, and sum(w)

McCullagh_compute_s_plus

Compute sums too use in maximizing log(likelihood)

Description

Compute sums too use in maximizing log(likelihood)

Usage

```
McCullagh_compute_s_plus(n)
```

Arguments

n	matrix of observed counts

Value

list of s_i_plus and s_plus_i

McCullagh_compute_update

Compute the Newton-Raphson update.

Description

Compute the Newton-Raphson update.

Usage

```
McCullagh_compute_update(gradient, hessian)
```

Arguments

gradient	gradient vector of log(likelihood) wrt parameters
hessian	hessian of log(likelihood) wrt parameters

Value

vector with update values for each of the parameters

`McCullagh_compute_z` *Computes Z, where z is w * lambda.*

Description

Computes Z, where z is w * lambda.

Usage

```
McCullagh_compute_z(lambda, w)
```

Arguments

`lambda`	cumulative logits
`w`	weights to apply to the logits

Value

z, sum pf product of lambda

`McCullagh_conditional_symmetry`
Fits the McCullagh (1978) conditional-symmetry model.

Description

McCullagh, P. (1978). A class of parametric models for the analysis of square contingency tables with ordered categories. Biometrika, 65(2) 413-418.

Usage

```
McCullagh_conditional_symmetry(n, max_iter = 5, verbose = FALSE)
```

Arguments

`n`	matrix of observed counts
`max_iter`	maximum number of iterations to maximize the log(likelihood)
`verbose`	should cycle-by-cycle info be printed. Default is FALSE.

Value

a list containing theta: the asymmetry parameter chisq: chi-square g_squared: likelihood ratio G^2 df: degrees of freedom

Examples

```
McCullagh_conditional_symmetry(vision_data)
```

McCullagh_conditional_symmetry_compute_s
Computes sums used in maximizing theta.

Description

Computes sums used in maximizing theta.

Usage

```
McCullagh_conditional_symmetry_compute_s(n)
```

Arguments

n	matrix of observed counts

Value

list with s_i_plus and s_plus-i

McCullagh_conditional_symmetry_initialize_phi
Initializes symmetry matrix phi

Description

Initializes symmetry matrix phi

Usage

```
McCullagh_conditional_symmetry_initialize_phi(M)
```

Arguments

M	the number of rows/columns in phi

Value

the phi matrix

McCullagh_conditional_symmetry_maximize_phi
Maximizes log(likelihood) wrt phi.

Description

Maximizes log(likelihood) wrt phi.

Usage

```
McCullagh_conditional_symmetry_maximize_phi(n)
```

Arguments

n	matrix of observed counts

Value

phi matrix

McCullagh_conditional_symmetry_maximize_theta
Maximizes the log(likelihood) wrt theta.

Description

Maximizes the log(likelihood) wrt theta.

Usage

```
McCullagh_conditional_symmetry_maximize_theta(n)
```

Arguments

n	matrix of observed counts

Value

value of asymmetry parameter theta

McCullagh_conditional_symmetry_pi

Computes model-based proportions.

Description

Computes model-based proportions.

Usage

```
McCullagh_conditional_symmetry_pi(phi, theta)
```

Arguments

phi	the symmetric matrix
theta	the asymmetry parameter

Value

matrix of model-based p-values

McCullagh_derivative_condition_wrt_psi

Derivative of the condition wrt psi[i, j].

Description

Derivative of the condition wrt psi[i, j].

Usage

```
McCullagh_derivative_condition_wrt_psi(i, j)
```

Arguments

i	first index of psi
j	second index of psi

Value

derivative

McCullagh_derivative_gamma_plus_1_wrt_phi

Derivative of gamma j + 1 wrt phi.

Description

Derivative of gamma j + 1 wrt phi.

Usage

```
McCullagh_derivative_gamma_plus_1_wrt_phi(gamma, j, phi)
```

Arguments

gamma	vector
j	index of gamma to take derivative of
phi	scalar phi taking derivative wrt

Value

derivative

McCullagh_derivative_gamma_wrt_phi

Derivative of gamma wrt phi.

Description

Version given in McCullagh isn't right.

Usage

```
McCullagh_derivative_gamma_wrt_phi(gamma, j, phi)
```

Arguments

gamma	vector of cumulative logits
j	index of derivative sought
phi	scalar phi taking derivative wrt

Value

derivative

McCullagh_derivative_gamma_wrt_y

Derivative of y wrt gamma.

Description

Assumes a logit link is being used.

Usage

```
McCullagh_derivative_gamma_wrt_y(gamma, i, j)
```

Arguments

gamma	matrix of gamma values
i	row index of gamma
j	column index of gamma

Value

derivative

McCullagh_derivative_lagrangian_wrt_delta

Derivative of Lagrange multiplier wrt scalar delta.

Description

Derivative of Lagrange multiplier wrt scalar delta.

Usage

```
McCullagh_derivative_lagrangian_wrt_delta(n, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
psi	symmetry matrix
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing coefficient so that sum o pi = 1. Default value is 1.0

Value

value of the derivative

McCullagh_derivative_lagrangian_wrt_delta_vec
Derivative of Lagrangian wrt delta_vec.

Description

Derivative of Lagrangian wrt delta_vec.

Usage

```
McCullagh_derivative_lagrangian_wrt_delta_vec(
  n,
  k,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
k	index of delta_vec to compute derivative wrt
psi	matrix of symmetry parameters
delta_vec	vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_derivative_lagrangian_wrt_psi
Derivative of Lagrangian wrt psi[i1, j1].

Description

Derivative of Lagrangian wrt psi[i1, j1].

Usage

```
McCullagh_derivative_lagrangian_wrt_psi(n, i1, j1, psi, delta, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`i1`	first index of psi
`j1`	first index of psi
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_derivative_log_l_wrt_alpha`

Derivative of log(likelihood) wrt alpha[index].

Description

Derivative of log(likelihood) wrt alpha[index].

Usage

```
McCullagh_derivative_log_l_wrt_alpha(n, index, psi, delta, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_derivative_log_l_wrt_beta`
Derivative of log(likelihood) wrt beta, as given in appendix of McCullagh.

Description

McCullagh, P. (1980). Regression models for ordinal data. Journal of the Royal Stastical Society, Series B, 42(2), 109-142. With assist from appendix of Agresti, (1984). Agresti, A. (1984). Analysis of ordinal categorical data. New York, Wiley, p. 244-246.

Usage

```
McCullagh_derivative_log_l_wrt_beta(n, x, gamma)
```

Arguments

`n`	matrix of observed counts
`x`	design matrix for location
`gamma`	matrix of model-based cumulative logits

Value

derivative

`McCullagh_derivative_log_l_wrt_c`
Derivative of log(likelihood) wrt c.

Description

Derivative of log(likelihood) wrt c.

Usage

```
McCullagh_derivative_log_l_wrt_c(n, psi, delta, alpha, c)
```

Arguments

`n`	matrix of observed counts
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_derivative_log_l_wrt_delta`
Derivative of log(likelihood) wrt delta (scalar or vector0.

Description

Derivative of log(likelihood) wrt delta (scalar or vector0.

Usage

```
McCullagh_derivative_log_l_wrt_delta(n, psi, delta, alpha, c = 1, k = 1)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.
k	index into delta_vac. Defaults to 1.

Value

derivative

`McCullagh_derivative_log_l_wrt_delta_vec`
Derivative of log(likelihood) wrt delta_vec[k].

Description

Derivative of log(likelihood) wrt delta_vec[k].

Usage

```
McCullagh_derivative_log_l_wrt_delta_vec(n, k, psi, delta_vec, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`k`	index of delta_vec
`psi`	matrix of symmetry parameters
`delta_vec`	vector asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_derivative_log_l_wrt_params`
Derivative of log(likelihood) wrt parameters.

Description

Derivative of log(likelihood) wrt parameters.

Usage

```
McCullagh_derivative_log_l_wrt_params(n, x, beta)
```

Arguments

`n`	matrix of observed counts
`x`	design matrix for location model
`beta`	vector of regression parameters for location model

Value

gradient vector

McCullagh_derivative_log_l_wrt_phi

Derivative of log(likelihood) wrt phi[i, j]

Description

Derivative of log(likelihood) wrt phi[i, j]

Usage

```
McCullagh_derivative_log_l_wrt_phi(n, phi, i, j)
```

Arguments

n	matrix of observed counts
phi	matrix of phi-values
i	row index of phi
j	column index of phi

Value

derivative

McCullagh_derivative_log_l_wrt_psi

Derivative of log(likelihood) wrt psi.

Description

Derivative of log(likelihood) wrt psi.

Usage

```
McCullagh_derivative_log_l_wrt_psi(n, i1, j1, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_derivative_omega_wrt_alpha

Derivative of Lagrange multiplier omega wrt alpha[index].

Description

Derivative of Lagrange multiplier omega wrt alpha[index].

Usage

```
McCullagh_derivative_omega_wrt_alpha(n, index, psi, delta, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_derivative_omega_wrt_c

Derivative of Lagrange multiplier omega wrt c.

Description

Derivative of Lagrange multiplier omega wrt c.

Usage

```
McCullagh_derivative_omega_wrt_c(n, psi, delta, alpha, c)
```

Arguments

`n`	matrix of observed counts
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_derivative_omega_wrt_delta` *Derivative of Lagrange multiplier omega wrt scalar delta.*

Description

Derivative of Lagrange multiplier omega wrt scalar delta.

Usage

```
McCullagh_derivative_omega_wrt_delta(n, psi, delta, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_derivative_omega_wrt_delta_vec
Derivative of Lagrange multiplier omega wrt vector delta[k].

Description

Derivative of Lagrange multiplier omega wrt vector delta[k].

Usage

```
McCullagh_derivative_omega_wrt_delta_vec(n, k, psi, delta_vec, alpha, c = 1)
```

Arguments

n	matrix of observed counts
k	index of delta_vec
psi	matrix of symmetry parameters
delta_vec	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_derivative_omega_wrt_psi
Derivative of Lagrange multiplier omega wrt psi[i, j].

Description

Derivative of Lagrange multiplier omega wrt psi[i, j].

Usage

```
McCullagh_derivative_omega_wrt_psi(n, i, j, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
i	first index of psi
j	second index of psi
psi	symmetry matrix
delta	scalar or vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Defaults to 1.0

McCullagh_derivative_phi_wrt_gamma

Derivative of phi wrt gamma.

Description

Derivative of phi wrt gamma.

Usage

```
McCullagh_derivative_phi_wrt_gamma(gamma, j)
```

Arguments

gamma	vector of gamma values
j	index of gamma for which to compute the derivative

Value

derivative

McCullagh_derivative_pi_wrt_alpha

Derivative of pi[i, j] wrt alpha[index].

Description

Derivative of pi[i, j] wrt alpha[index].

Usage

```
McCullagh_derivative_pi_wrt_alpha(i, j, index, psi, delta, alpha, c = 1)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`index`	index of alpha
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_derivative_pi_wrt_c

Derivative pi[i, j] wrt c.

Description

Derivative pi[i, j] wrt c.

Usage

```
McCullagh_derivative_pi_wrt_c(i, j, psi, delta, alpha, c)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`psi`	the matrix of symmetry parameters
`delta`	the scalar or vector asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0

Value

derivative

McCullagh_derivative_pi_wrt_delta

Derivative of pi[i, j] wrt delta.

Description

Derivative of pi[i, j] wrt delta.

Usage

```
McCullagh_derivative_pi_wrt_delta(i, j, psi, delta, alpha, c = 1)
```

Arguments

i	row index of pi
j	column index of pi
psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_derivative_pi_wrt_delta_vec

Derivative pi[i, j] wrt delta[k].

Description

Derivative pi[i, j] wrt delta[k].

Usage

```
McCullagh_derivative_pi_wrt_delta_vec(i, j, k, psi, delta_vec, alpha, c = 1)
```

Arguments

i	row index of pi
j	column index of pi
k	index of delta_vec
psi	the matrix of symmetry parameters
delta_vec	the vector asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

`McCullagh_derivative_pi_wrt_psi`
Derivative of pi[i, j] wrt psi[i1, j1].

Description

Derivative of pi[i, j] wrt psi[i1, j1].

Usage

```
McCullagh_derivative_pi_wrt_psi(i, j, i1, j1, psi, delta, alpha, c = 1)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`i1`	row index of psi
`j1`	column index of psi
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

`McCullagh_derivative_pij_wrt_alpha`
Derivative of pij[i, j] wrt alpha[index]

Description

Derivative of pij[i, j] wrt alpha[index]

Usage

```
McCullagh_derivative_pij_wrt_alpha(i, j, index, psi, delta, alpha, c = 1)
```

Arguments

`i`	row index of pij
`j`	column index of pij
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar or vector of asymmetry parameters
`alpha`	vector of asymmetry parameters
`c`	normalizing constant to make pi sum to 1.0. Default ot 1.0

Value

derivative

`McCullagh_derivative_pij_wrt_c`
Derivative pij[i, j] wrt c.

Description

Derivative pij[i, j] wrt c.

Usage

```
McCullagh_derivative_pij_wrt_c(i, j, psi, delta, alpha, c)
```

Arguments

`i`	row index of pij
`j`	column index of pij
`psi`	matrix of symmetry parameters
`delta`	scalar or vector of asymmetry parameters
`alpha`	vector of asymmetry parameters
`c`	normalizing constant to make pi sum to 1.0

Value

derivative

`McCullagh_derivative_pij_wrt_delta`
Derivative of pij[i, j] wrt scalar delta.

Description

Derivative of pij[i, j] wrt scalar delta.

Usage

```
McCullagh_derivative_pij_wrt_delta(i, j, psi, delta, alpha, c = 1)
```

Arguments

i	row index of pij
j	column index of pij
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing constant so that pi sum to 1.0. Default value is 1.0

Value

derivative

`McCullagh_derivative_pij_wrt_delta_vec`
Derivative pij[i,j] wrt vector delta[k].

Description

Derivative pij[i,j] wrt vector delta[k].

Usage

```
McCullagh_derivative_pij_wrt_delta_vec(i, j, k, psi, delta_vec, alpha, c = 1)
```

Arguments

i	row index of pij
j	column index of pij
k	index of delta
psi	the matrix of symmetry parameters
delta_vec	the vector asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

list containing matrices pij and qij

McCullagh_derivative_pij_wrt_psi
Derivative of pij[a, b] wrt psi[h, k]

Description

Derivative of pij[a, b] wrt psi[h, k]

Usage

```
McCullagh_derivative_pij_wrt_psi(a, b, h, k, delta, alpha, c = 1)
```

Arguments

a	row index of pi
b	column index of pi
h	row index of phi
k	column index of phi
delta	scalar or vector version of asymmetry parameters
alpha	vector of asymmetry parameters
c	normalizing constant for to make pi sum to 1. Defaults to 1.0

Value

derivative

McCullagh_extract_weights
Extracts the weights to convert cumulative model-based probabilities to regular probabilities.

Description

Extracts the weights to convert cumulative model-based probabilities to regular probabilities.

Usage

```
McCullagh_extract_weights(i, j, M)
```

Arguments

`i`	row index sought
`j`	column index sought
`M`	the number of rows/columns in observed matrix

Value

a list containing w_psi for when i == j w_pij for when i < j w_qij for when j < i weight populated with correct entry based on actual i and j

---

`McCullagh_fit_location_regression_model` *Fit location model*

Description

Fit location model

Usage

```
McCullagh_fit_location_regression_model(n, x, max_iter = 5, verbose = FALSE)
```

Arguments

`n`	matrix of observed counts
`x`	design matrix for regression model
`max_iter`	maximum number of Fisher scoring iterations
`verbose`	logical: should cycle-by-cycle info be printed out? Default value is FALSE, do not print

Value

a list containing beta: regression parameter estimates se: matrix of estimated standard errors cov: covariance matrix of parameter estimates g_squared: G^2 likelihood ratio chi-square for model chisq: Pearson chi-square for model df: degrees of freedom

McCullagh_generalized_palindromic_symmetry
Generalized version of palindromic symmetry model

Description

delta now is a vector, varying by index McCullagh, P. (1978). A class of parametric models for the analysis of square contingency tables with ordered categories. Biometrika, 65(2). 413-416.

Usage

```
McCullagh_generalized_palindromic_symmetry(
  n,
  max_iter = 15,
  verbose = FALSE,
  start_values = FALSE
)
```

Arguments

n	matrix of observed counts
max_iter	maximum number of iterations to maximize log(likelihood)
verbose	should cycle-by-cycle information be printed out? Default is FALSE, do not print
start_values	logical should the regular palindomic symmetry model be fit first to get good starting values. Default is FALSE.

Value

a list containing

a list containing delta: the vector of asymmetry parameter delta sigma_delta: vector of SE(delta) logL: value of log(likelihood) for final estimates chisq: Pearson chi-square for solution df: degrees of freedom for solution chisq psi: matrix of symmetry parameters alpha: c: constraint, sum of pi - values condition: constraint on psi to make model identified, Lagrange multiplier SE: vector of standard errors for all parameters

Examples

```
McCullagh_generalized_palindromic_symmetry(vision_data)
```

McCullagh_generalized_pij_qij
Computes culuative model probabilities for the generalized model using vector delta.

Description

Computes culuative model probabilities for the generalized model using vector delta.

Usage

```
McCullagh_generalized_pij_qij(i, j, psi, delta_vec, alpha, c1 = 1)
```

Arguments

i	row index
j	column index
psi	symmetry matrix
delta_vec	vector of delta values
alpha	vector of asymmetry values
c1	normalizing value for pi. Defaults to 1.0

Value

model-based cumulative probability pi_ij

McCullagh_generate_names
Generates names to label the parameters.

Description

Generates names to label the parameters.

Usage

```
McCullagh_generate_names(psi, delta, alpha, c)
```

Arguments

psi	matrix of symmetry parameters
delta	scalar of matrix of asymmetry parameters
alpha	vector of asymmetry parameters
c	scling factor to ensure sup of pi is 1.0

Value

character vector of labels for the SE values

McCullagh_get_statistics

Computes summary statistics needed to compute estimate of delta.

Description

Computes summary statistics needed to compute estimate of delta.

Usage

```
McCullagh_get_statistics(m)
```

Arguments

`m`	matrix of observed counts

Value

a list containing: N: matrix of sums above and below the diagonal n: vector, size of binomial r: vector, observed sums, number of successes for binomail

---

McCullagh_gradient_log_l

Gradient vector of log(likelihood)

Description

Gradient vector of log(likelihood)

Usage

```
McCullagh_gradient_log_l(n, psi, delta, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`psi`	matrix of symmetry parameters
`delta`	scalar or vector asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

gradient vector of first-order partials wrt log(likelihood0)

McCullagh_hessian_log_l

Hessian matrix of log(likelihood)

Description

Hessian matrix of log(likelihood)

Usage

```
McCullagh_hessian_log_l(n, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar or vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

hessian matrix of second-order partials wrt log(likelihood0)

McCullagh_initialize_beta

Initializes the beta vector.

Description

Initializes the beta vector.

Usage

```
McCullagh_initialize_beta(n, c, v)
```

Arguments

n	matrix of observed counts
c	number of score levels in table
v	number of levels of beta beyond c

Value

initialized beta vector

McCullagh_initialize_delta

Compute initial values for scalar delta

Description

Compute initial values for scalar delta

Usage

```
McCullagh_initialize_delta(n)
```

Arguments

n	matrix of observed counts

Value

value of delta

McCullagh_initialize_delta_vec

Initialize vector delta

Description

Initialize vector delta

Usage

```
McCullagh_initialize_delta_vec(n)
```

Arguments

n	matrix of observed counts

Value

vector of delta values

McCullagh_initialize_psi

Initialize the symmetry matrix psi

Description

Initialize the symmetry matrix psi

Usage

```
McCullagh_initialize_psi(n, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
delta	scalar delta value
alpha	vector of asymmetry parameters
c	normalizing value of pi. Default is 1.0

Value

matrix psi

McCullagh_initialize_x

Initialize design matrix for location model.

Description

This is the simplest possible implementation, that fits thresholds and a single group contrast. More complex problems will implement the matrix X themselves.

Usage

```
McCullagh_initialize_x(s, c, v)
```

Arguments

s	number of levels of stratification variable
c	number of score levels
v	number of predictors above thresholds

Value

design matrix X

McCullagh_is_in_constraint_set

Logical test of whether a specific psi will be in the constraint set.

Description

Logical test of whether a specific psi will be in the constraint set.

Usage

```
McCullagh_is_in_constraint_set(i, j)
```

Arguments

i	first index of psi
j	second index of psi

Value

TRUE if it falls within the set, FALSE otherwise.

McCullagh_is_pi_invalid

Test whether pi matrix is valid, i.e., 0 < all values.

Description

Test whether pi matrix is valid, i.e., 0 < all values.

Usage

```
McCullagh_is_pi_invalid(pi)
```

Arguments

pi	matrix of pi values to be tested.

Value

TRUE if all pi > 0, FALSE otherwise.

McCullagh_log_L — *Computes the log(likelihood).*

Description

Computes the log(likelihood).

Usage

```
McCullagh_log_L(n, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar or vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_logistic_model — *MCCullagh's logistic model.*

Description

McCullah, P. (1977). A logistic model for paired comparisons with ordered categorical data. Biometrika, 64(3), 449-453.

Usage

```
McCullagh_logistic_model(m)
```

Arguments

m	matrix of observed counts

Value

a list containing w_tilde: vector of model weights for sum of normally distributed components delta_tilde: delta parameter computed using w_tilde w_star: vector of weights for Mantel-Haenszel type numerator and denominator delta_star: delta parameter computed using w_star var: variance of delta estimate

Examples

```
McCullagh_logistic_model(coal_g)
```

McCullagh_logits *Computed cumulative logits.*

Description

Computed cumulative logits.

Usage

```
McCullagh_logits(cumulative, use_half = TRUE)
```

Arguments

cumulative	vector of cumulative counts
use_half	logical indicting whether or not to add 0.5 to numerator and denominator counts before computing logits, Default value is TRUE, add 0.5.

McCullagh_maximize_q_symmetry *Maximize the log(likelihood) wrt parameters phi and alpha*

Description

Maximize the log(likelihood) wrt parameters phi and alpha

Usage

```
McCullagh_maximize_q_symmetry(n, phi, alpha)
```

Arguments

n	matrix of observed counts
phi	matrix of symmetry parameters
alpha	vector of asymmetry parameters

Value

list with new values of phi and alpha

McCullagh_newton_raphson_update
Newton-Raphson update.

Description

Using gradient and hessian, it finds the update direction. Then it tries increassingly smaller step sizes until the step*update yields a valid pi matrix.

Usage

```
McCullagh_newton_raphson_update(
  n,
  gradient,
  hessian,
  psi,
  delta,
  alpha,
  c = 1,
  max_iter = 50,
  verbose = FALSE
)
```

Arguments

n	matrix of observed counts
gradient	gradient vector
hessian	hessian matrix
psi	matrix of symmetry parameters
delta	scalar or vector of asymmetry parameters
alpha	vector of asymmetry parameters
c	scaling factor to ensure pi sums to 1.0. Default is 1.0
max_iter	maximum number of iterations. Default is 50.
verbose	should cycle-by-cycle into be printed out. Default is FALSE, do not print.

Value

list containing new parameters psi: matrix of symmetry parameters delta; scalar or vector of asymmetry parameters alpha: vector of asymmetry parameters c: scaling coefficient to ensure pi sums to 1.0

McCullagh_palindromic_symmetry
McCullagh's palindromic symmetry model

Description

McCullagh, P. (1978). A class of parametric models for the analysis of square contingency tables with ordered categories. Biometrika, 65(2). 413-416.

Usage

```
McCullagh_palindromic_symmetry(n, max_iter = 15, verbose = FALSE)
```

Arguments

n	matrix of observed counts
max_iter	maximum number of iterations to maximize the log(likelihood)
verbose	should cycle-by-cycle info be printed out? Default is FALSE, don't print.

Value

a list containing delta: the value of the asymmetry parameter delta sigma_delta: SE(delta) logL: value of log(likelihood) for final estimates chisq: Pearson chi-square for solution df: degrees of freedom for solution chisq psi: matrix of symmetry parameters alpha: c: constraint, sum of pi - values condition: constraint on psi to make model identified, Lagrange multiplier SE: vector of standard errors for all parameters

Examples

```
McCullagh_palindromic_symmetry(vision_data)
```

McCullagh_penalized *Computes the penalized value of a derivative by adding the derivative of the penalty to it.*

Description

Computes the penalized value of a derivative by adding the derivative of the penalty to it.

Usage

```
McCullagh_penalized(derivative, i1, j1, n, psi, delta, alpha, c = 1)
```

Arguments

`derivative`	the base derivative
`i1`	first index of psi
`j1`	second index of psi
`n`	matrix of observed counts
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_pij_qij` *Compute model-based cumulative probabilities*

Description

Compute model-based cumulative probabilities

Usage

```
McCullagh_pij_qij(i, j, psi, delta, alpha, c = 1)
```

Arguments

`i`	row index
`j`	column index
`psi`	the symmetry matrix
`delta`	the asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for pi. Default is 1.0

Value

the model-based cumulative probability pi_ij

McCullagh_proportional_hazards
Computes the proportional hazards.

Description

Computes the proportional hazards.

Usage

```
McCullagh_proportional_hazards(n)
```

Arguments

n	matrix of observed counts

Value

loga(-log(survival))

McCullagh_q_symmetry_initialize_alpha
Initializes the asymmetry vector alpha

Description

Initializes the asymmetry vector alpha

Usage

```
McCullagh_q_symmetry_initialize_alpha(M)
```

Arguments

M	size of alpha vector to create = nrow(matrix to analyze)

Value

vector of asymmetry parameters alpha

McCullagh_q_symmetry_initialize_phi

Initializes the phi matrix

Description

Initializes the phi matrix

Usage

```
McCullagh_q_symmetry_initialize_phi(M)
```

Arguments

M	size of the psi matrix to create

Value

the symmetry matrix phi

McCullagh_q_symmetry_pi

Computes the model-based p-values

Description

Computes the model-based p-values

Usage

```
McCullagh_q_symmetry_pi(phi, alpha)
```

Arguments

phi	the matrix of symmetry parameters
alpha	the vector of asymmetry parameters

Value

matrix pi of model-based p-values

McCullagh_quasi_symmetry

Fits McCullagh's (1978) quasi-symmetry model.

Description

McCullagh, P. (1978). A class of parametric models for the analysis of square contingency tables with ordered categories. Biometrika, 65(2) 413-418.

Usage

```
McCullagh_quasi_symmetry(n, max_iter = 15, verbose = FALSE)
```

Arguments

n	matrix of observed counts
max_iter	maximum number of iterations in maximizing log(likelihood), Default is 15.
verbose	should cycle-by-cycle information be printed out? Default is FALSE, do not print

Value

a list containing phi: symmetry matrix alpha: vector of asymmetry parameters chisq: Pearson chi-square value df; degrees of freedom

Examples

```
McCullagh_quasi_symmetry(vision_data)
```

McCullagh_second_order_lagrangian_wrt_psi_2

Second derivative of Lagrangian wrt psi^2.

Description

Second derivative of Lagrangian wrt psi^2.

Usage

```
McCullagh_second_order_lagrangian_wrt_psi_2(
  n,
  i1,
  j1,
  i2,
  j2,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	first row index of psi
j1	first column index of psi
i2	second row index of psi
j2	second column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_lagrangian_wrt_psi_alpha

Second derivative of Lagrangian wrt psi[i1, j1] and alpha[index].

Description

Second derivative of Lagrangian wrt psi[i1, j1] and alpha[index].

Usage

```
McCullagh_second_order_lagrangian_wrt_psi_alpha(
  n,
  i1,
  j1,
  index,
```

```
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
index	second row index of alpha
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_lagrangian_wrt_psi_delta

Second derivative of Lagrangian wrt psi[i1, j1] and delta.

Description

Second derivative of Lagrangian wrt psi[i1, j1] and delta.

Usage

```
McCullagh_second_order_lagrangian_wrt_psi_delta(
  n,
  i1,
  j1,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_second_order_lagrangian_wrt_psi_delta_vec`
Second derivative of Lagrangian wrt psi[i1, j1] and delta_vec[k[.

Description

Second derivative of Lagrangian wrt psi[i1, j1] and delta_vec[k[.

Usage

```
McCullagh_second_order_lagrangian_wrt_psi_delta_vec(
  n,
  i1,
  j1,
  k,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
k	index of delta_vec
psi	matrix of symmetry parameters
delta_vec	vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_second_order_log_l_wrt_alpha_2`
Second derivative of log(likelihood) wrt alpha^2.

Description

Second derivative of log(likelihood) wrt alpha^2.

Usage

```
McCullagh_second_order_log_l_wrt_alpha_2(
  n,
  index_a,
  index_b,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
index_a	first index of alpha
index_b	second column index of alpha
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_alpha_c
Second derivative of log(likelihood) wrt alpha[index] and c.

Description

Second derivative of log(likelihood) wrt alpha[index] and c.

Usage

```
McCullagh_second_order_log_l_wrt_alpha_c(n, index, psi, delta, alpha, c)
```

Arguments

`n`	matrix of observed counts
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_beta_2
Expected values of second order derivatives of log(likelihood) wrt beta.

Description

Appendix of McCullagh, P. (1980). Regression models for ordinal data. Journal of the Royal Statistical Society, Series B, 42(2), 109-142. and appendix B3 of Agresti, A. (1984). Analysis of ordinal categorical data, New York, Wiley, p. 242-244.

Usage

```
McCullagh_second_order_log_l_wrt_beta_2(n, x, gamma)
```

Arguments

`n`	matrix of observed counts
`x`	design matrix for location model
`gamma`	current value of model-based cumulative logits.

Value

matrix of second order partial derivatives

McCullagh_second_order_log_l_wrt_c_2

Second derivative of log(likelihood) wrt c^2.

Description

Second derivative of log(likelihood) wrt c^2.

Usage

```
McCullagh_second_order_log_l_wrt_c_2(n, psi, delta, alpha, c)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_delta_2

Second derivative of log(likelihood) wrt delta^2.

Description

Second derivative of log(likelihood) wrt delta^2.

Usage

```
McCullagh_second_order_log_l_wrt_delta_2(n, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_second_order_log_l_wrt_delta_alpha`
Second derivative of log(likelihood) wrt delta and alpha[index].

Description

Second derivative of log(likelihood) wrt delta and alpha[index].

Usage

```
McCullagh_second_order_log_l_wrt_delta_alpha(
  n,
  index,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

`n`	matrix of observed counts
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_delta_c
Second derivative of log(likelihood) wrt scalar delta and c.

Description

Second derivative of log(likelihood) wrt scalar delta and c.

Usage

```
McCullagh_second_order_log_l_wrt_delta_c(n, psi, delta, alpha, c)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0..

Value

derivative

McCullagh_second_order_log_l_wrt_delta_vec_2
Second derivative of log(likelihood) wrt delta_vec^2.

Description

Second derivative of log(likelihood) wrt delta_vec^2.

Usage

```
McCullagh_second_order_log_l_wrt_delta_vec_2(
  n,
  k1,
  k2,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

`n`	matrix of observed counts
`k1`	first index of delta_vec
`k2`	second index of delta_vec
`psi`	matrix of symmetry parameters
`delta_vec`	vector asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_delta_vec_alpha

Second derivative of log(likelihood) wrt delta[k] and alpha[index].

Description

Second derivative of log(likelihood) wrt delta[k] and alpha[index].

Usage

```
McCullagh_second_order_log_l_wrt_delta_vec_alpha(
  n,
  k,
  index,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

`n`	matrix of observed counts
`k`	index of delta_vec
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta_vec`	vector asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_second_order_log_l_wrt_delta_vec_c`
Second derivative of log(likeloihood) wrt delta_vec[k] and c.

Description

Second derivative of log(likeloihood) wrt delta_vec[k] and c.

Usage

```
McCullagh_second_order_log_l_wrt_delta_vec_c(n, k, psi, delta_vec, alpha, c)
```

Arguments

`n`	matrix of observed counts
`k`	index of delta_vec
`psi`	matrix of symmetry parameters
`delta_vec`	vector asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0

Value

derivative

`McCullagh_second_order_log_l_wrt_parms`
Expected second order derivatives of log(likelihood)

Description

Expected second order derivatives of log(likelihood)

Usage

```
McCullagh_second_order_log_l_wrt_parms(n, x, beta)
```

Arguments

`n`	matrix of observed counts
`x`	design matrix for location model
`beta`	vector of regression parameters for location model

Value

matrix of expected second derivatives

McCullagh_second_order_log_l_wrt_psi_2
Second derivative of log(likelihoood) wrt psi^2.

Description

Second derivative of log(likelihoood) wrt psi^2.

Usage

```
McCullagh_second_order_log_l_wrt_psi_2(
  n,
  i1,
  j1,
  i2,
  j2,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	first row index of psi
j1	first column index of psi
i2	second row index of psi
j2	second column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_psi_alpha
Second derivative of log(likelihoood) wrt ps[i1, j1] and alpha[index].

Description

Second derivative of log(likelihoood) wrt ps[i1, j1] and alpha[index].

Usage

```
McCullagh_second_order_log_l_wrt_psi_alpha(
  n,
  i1,
  j1,
  index,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
index	index of alpha
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_psi_c

Second derivative of log(likelihood) wrt psi[i1, j1] and c.

Description

Second derivative of log(likelihood) wrt psi[i1, j1] and c.

Usage

```
McCullagh_second_order_log_l_wrt_psi_c(n, i1, j1, psi, delta, alpha, c)
```

Arguments

`n`	matrix of observed counts
`i1`	row index of psi
`j1`	column index of psi
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_psi_delta

Second derivative of log(likelihood) wrt psi[i1, j1] and scalar delta..

Description

Second derivative of log(likelihood) wrt psi[i1, j1] and scalar delta..

Usage

```
McCullagh_second_order_log_l_wrt_psi_delta(n, i1, j1, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_log_l_wrt_psi_delta_vec

Second derivative of log(likelihood) wrt psi[i1, j1] and delta_vec[k].

Description

Second derivative of log(likelihood) wrt psi[i1, j1] and delta_vec[k].

Usage

```
McCullagh_second_order_log_l_wrt_psi_delta_vec(
  n,
  i1,
  j1,
  k,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
k	second row index of delta
psi	matrix of symmetry parameters
delta_vec	vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_second_order_omega_wrt_alpha_2`
Second derivative of Lagrange multiplier omega wrt alpha^2.

Description

Second derivative of Lagrange multiplier omega wrt alpha^2.

Usage

```
McCullagh_second_order_omega_wrt_alpha_2(n, k1, k2, psi, delta, alpha, c = 1)
```

Arguments

`n`	matrix of observed counts
`k1`	first index of alpha
`k2`	second index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

`McCullagh_second_order_omega_wrt_alpha_c`
Second derivative of Lagrange multiplier omega wrt alpha[index] and c.

Description

Second derivative of Lagrange multiplier omega wrt alpha[index] and c.

Usage

```
McCullagh_second_order_omega_wrt_alpha_c(n, index, psi, delta, alpha, c)
```

Arguments

n	matrix of observed counts
index	row index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_c_2

Second derivative of Lagrange multiplier omega wrt c^2.

Description

Second derivative of Lagrange multiplier omega wrt c^2.

Usage

```
McCullagh_second_order_omega_wrt_c_2(n, psi, delta, alpha, c)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_delta_2

Second derivative of Lagrange multiplier omega wrt scalae delta^2.

Description

Second derivative of Lagrange multiplier omega wrt scalae delta^2.

Usage

```
McCullagh_second_order_omega_wrt_delta_2(n, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_delta_alpha

Second derivative of Lagrange multiplier omega wrt delta and alpha[index].

Description

Second derivative of Lagrange multiplier omega wrt delta and alpha[index].

Usage

```
McCullagh_second_order_omega_wrt_delta_alpha(
  n,
  index,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

`n`	matrix of observed counts
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_delta_c

Second derivative of Lagrange multiplier omega wrt scalar delta and c.

Description

Second derivative of Lagrange multiplier omega wrt scalar delta and c.

Usage

```
McCullagh_second_order_omega_wrt_delta_c(n, psi, delta, alpha, c)
```

Arguments

`n`	matrix of observed counts
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_delta_vec_2

Second derivative of Lagrange multiplier omega wrt delta_vec^2.

Description

Second derivative of Lagrange multiplier omega wrt delta_vec^2.

Usage

```
McCullagh_second_order_omega_wrt_delta_vec_2(
  n,
  k1,
  k2,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
k1	first index of delta_vec
k2	second index of delta_vec
psi	matrix of symmetry parameters
delta_vec	vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_delta_vec_alpha

Second derivative of Lagrange multiplier omega wrt delta_vec[k] and alpha[index].

Description

Second derivative of Lagrange multiplier omega wrt delta_vec[k] and alpha[index].

Usage

```
McCullagh_second_order_omega_wrt_delta_vec_alpha(
  n,
  k,
  index,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
k	index of delta_vec
index	index of alpha
psi	matrix of symmetry parameters
delta_vec	vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_delta_vec_c

Second derivative of Lagrange multiplier omega wrt delta_vec[k] and c.

Description

Second derivative of Lagrange multiplier omega wrt delta_vec[k] and c.

Usage

```
McCullagh_second_order_omega_wrt_delta_vec_c(n, k, psi, delta_vec, alpha, c)
```

Arguments

n	matrix of observed counts
k	index of delta_vec
psi	matrix of symmetry parameters
delta_vec	vector of asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_psi_2
Second derivative of Lagrange multiplier omega wrt psi^2.

Description

Second derivative of Lagrange multiplier omega wrt psi^2.

Usage

```
McCullagh_second_order_omega_wrt_psi_2(
  n,
  i1,
  j1,
  i2,
  j2,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	first row index of psi
j1	first column index of psi
i2	second row index of psi
j2	second column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_psi_alpha
Second derivative of Lagrange multiplier omega wrt psi[i1, j1] and alpha[index].

Description

Second derivative of Lagrange multiplier omega wrt psi[i1, j1] and alpha[index].

Usage

```
McCullagh_second_order_omega_wrt_psi_alpha(
  n,
  i1,
  j1,
  index,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

`n`	matrix of observed counts
`i1`	row index of psi
`j1`	column index of psi
`index`	index of alpha
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_psi_c

Second derivative of Lagrange multiplier omega wrt psi[i1, j1] and c.

Description

Second derivative of Lagrange multiplier omega wrt psi[i1, j1] and c.

Usage

```
McCullagh_second_order_omega_wrt_psi_c(n, i1, j1, psi, delta, alpha, c)
```

Arguments

`n`	matrix of observed counts
`i1`	row index of psi
`j1`	column index of psi
`psi`	matrix of symmetry parameters
`delta`	scalar asymmetry parameter
`alpha`	vector of asymmetry parameters
`c`	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_psi_delta

Second derivative of Lagrange multiplier omega wrt psi and scalar delta.

Description

Second derivative of Lagrange multiplier omega wrt psi and scalar delta.

Usage

```
McCullagh_second_order_omega_wrt_psi_delta(n, i1, j1, psi, delta, alpha, c = 1)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
psi	matrix of symmetry parameters
delta	scalar asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_omega_wrt_psi_delta_vec

Second derivative of Lagrange multiplier omega wrt psi[i1, j1] and delta_vec[k].

Description

Second derivative of Lagrange multiplier omega wrt psi[i1, j1] and delta_vec[k].

Usage

```
McCullagh_second_order_omega_wrt_psi_delta_vec(
  n,
  i1,
  j1,
  k,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

n	matrix of observed counts
i1	row index of psi
j1	column index of psi
k	index of delta_vec
psi	matrix of symmetry parameters
delta_vec	vector asymmetry parameter
alpha	vector of asymmetry parameters
c	normalizing factor to make pi sum to 1.0. Default is 1.0.

Value

derivative

McCullagh_second_order_pi_wrt_alpha_2
Second derivative of pi[i, j] wrt alpha^2.

Description

Second derivative of pi[i, j] wrt alpha^2.

Usage

```
McCullagh_second_order_pi_wrt_alpha_2(
  i,
  j,
  index1,
  index2,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

i	row index of pi
j	column index of pi
index1	index of first alpha
index2	index of second aloha
psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_second_order_pi_wrt_alpha_c

Second derivaitve of pi[i, j] wrt alpha[index] and c.

Description

Second derivaitve of pi[i, j] wrt alpha[index] and c.

Usage

```
McCullagh_second_order_pi_wrt_alpha_c(i, j, index, psi, delta, alpha, c)
```

Arguments

i	row index of pi
j	column index of pi
index	index of alpha
psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0

Value

derivative

McCullagh_second_order_pi_wrt_c_2

Second order derivative of pi[i, j] wrt c^2.

Description

Second order derivative of pi[i, j] wrt c^2.

Usage

```
McCullagh_second_order_pi_wrt_c_2(i, j, psi, delta, alpha, c)
```

Arguments

i	row index of pi
j	column index of pi
psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

`McCullagh_second_order_pi_wrt_delta_2`
Second order derivative of pi[i, j] wrt scalar delta.

Description

Second order derivative of pi[i, j] wrt scalar delta.

Usage

```
McCullagh_second_order_pi_wrt_delta_2(i, j, psi, delta, alpha, c = 1)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

`McCullagh_second_order_pi_wrt_delta_alpha`
Second order deriviative of pi[i, j] wrt scalar delta and alpha[index]

Description

Second order deriviative of pi[i, j] wrt scalar delta and alpha[index]

Usage

```
McCullagh_second_order_pi_wrt_delta_alpha(
  i,
  j,
  index,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`index`	index of alpha
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

`McCullagh_second_order_pi_wrt_delta_c`
Second order derivative of pi[i, j] wrt scalae delta and c.

Description

Second order derivative of pi[i, j] wrt scalae delta and c.

Usage

```
McCullagh_second_order_pi_wrt_delta_c(i, j, psi, delta, alpha, c)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0

Value

derivative

McCullagh_second_order_pi_wrt_delta_vec_2
Derivative of pi[i, j] wrt delta^2.

Description

Derivative of pi[i, j] wrt delta^2.

Usage

```
McCullagh_second_order_pi_wrt_delta_vec_2(
  i,
  j,
  k1,
  k2,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

i	row index of pi
j	column index of pi
k1	first index of delta
k2	second index of delta
psi	the matrix of symmetry parameters
delta_vec	the vector asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_second_order_pi_wrt_delta_vec_alpha

Second order dertivative of pi[i, j] wrtt delta[k] alpha[index].

Description

Second order dertivative of pi[i, j] wrtt delta[k] alpha[index].

Usage

```
McCullagh_second_order_pi_wrt_delta_vec_alpha(
  i,
  j,
  k,
  index,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

i	row index of pi
j	column index of pi
k	index of delta
index	index of alpha
psi	the matrix of symmetry parameters
delta_vec	the vector asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_second_order_pi_wrt_delta_vec_c

Second derivative of pi[i, j] wrt delta[k] and c.

Description

Second derivative of pi[i, j] wrt delta[k] and c.

Usage

```
McCullagh_second_order_pi_wrt_delta_vec_c(i, j, k, psi, delta_vec, alpha, c)
```

Arguments

i	row index of pi
j	column index of pi
k	index of delta
psi	the matrix of symmetry parameters
delta_vec	the vector asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_second_order_pi_wrt_psi_2

Second order derivative wrt psi^2.

Description

Second order derivative wrt psi^2.

Usage

```
McCullagh_second_order_pi_wrt_psi_2(
  i,
  j,
  i1,
  j1,
  i2,
  j2,
  psi,
```

```
  delta,
  alpha,
  c = 1
)
```

Arguments

i	row index of pi
j	column index of pi
i1	first row index of psi
j1	first column index of psi
i2	second row index of psi
j2	second column index of pis
psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_second_order_pi_wrt_psi_alpha
Second order derivative of pi[i, j] wrt psi[i1, j1] and alpha[index].

Description

Second order derivative of pi[i, j] wrt psi[i1, j1] and alpha[index].

Usage

```
McCullagh_second_order_pi_wrt_psi_alpha(
  i,
  j,
  i1,
  j1,
  index,
  psi,
  delta,
  alpha,
  c = 1
)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`i1`	row index of psi
`j1`	column index of psi
`index`	index of alpha
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

`McCullagh_second_order_pi_wrt_psi_c`
Second order derivative of pi[i, j] wrt psi[i1, j1] and c.

Description

Second order derivative of pi[i, j] wrt psi[i1, j1] and c.

Usage

```
McCullagh_second_order_pi_wrt_psi_c(i, j, i1, j1, psi, delta, alpha, c)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`i1`	row index of psi
`j1`	column index of psi
`psi`	the matrix of symmetry parameters
`delta`	the scalar asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0

Value

derivative

McCullagh_second_order_pi_wrt_psi_delta

Second order derivaitve of pi wrt pshi and scalar delta.

Description

Second order derivaitve of pi wrt pshi and scalar delta.

Usage

```
McCullagh_second_order_pi_wrt_psi_delta(i, j, i1, j1, psi, delta, alpha, c = 1)
```

Arguments

i	row index of pi
j	column index of pi
i1	row index of psi
j1	column index of psi
psi	the matrix of symmetry parameters
delta	the scalar asymmetry parameter
alpha	the vector of asymmetry parameters
c	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_second_order_pi_wrt_psi_delta_vec

Second order derivaitve of pi[i, j] wrt psi[i1, j1] and kelta[k].

Description

Second order derivaitve of pi[i, j] wrt psi[i1, j1] and kelta[k].

Usage

```
McCullagh_second_order_pi_wrt_psi_delta_vec(
  i,
  j,
  i1,
  j1,
  k,
  psi,
  delta_vec,
  alpha,
  c = 1
)
```

Arguments

`i`	row index of pi
`j`	column index of pi
`i1`	row index of psi
`j1`	column index of psi
`k`	index of delta
`psi`	the matrix of symmetry parameters
`delta_vec`	the vector asymmetry parameter
`alpha`	the vector of asymmetry parameters
`c`	the normalizing constant for the pis to sum to 1.0 Default value is 1.0

Value

derivative

McCullagh_update_parameters

Update the parameters based on Newton-Raphson step.

Description

Update the parameters based on Newton-Raphson step.

Usage

```
McCullagh_update_parameters(update, step, psi, delta, alpha, c = 1)
```

Arguments

`update`	vector of update values
`step`	size of candidate step along direction of update
`psi`	vector of symmetry parameters
`delta`	scalar or vector of asymmetry parameters
`alpha`	vector of asymmetry parameters
`c`	normalization factor to make sum pf pi = 1.0. Default value is 1.0.

Value

list containing new parameters psi: matrix of symmetry parameters delta; scalar or vector of asymmetry parameters alpha: vector of asymmetry parameters c: scaling coefficient to ensure pi sums to 1.0

`McCullagh_v_inverse` *Compute v_inverse (from appendix).*

Description

Compute v_inverse (from appendix).

Usage

```
McCullagh_v_inverse(gamma, i, j)
```

Arguments

`gamma`	matrix of cumulative logits
`i`	row index
`j`	column index

Value

V^(-1) : d phi / d gamma[i, j]

mental_health	*Relationship between child's mental health and parents' socioeconomic status.*

Description

Rows are child's mental health (ranging from 1 = well to 4 = impaired), and columns are parents' socioeconomic status, A - F.

Usage

```
mental_health
```

Format

'mental_health' A matrix with 4 rows and 6 columns

Source

Goodman, L. A. (1979). Simple models for the analysis of association in cross-classifications having ordered categories.

model_i_column_theta	*Computes the column association values theta-hat*

Description

Computes the column association values theta-hat

Usage

```
model_i_column_theta(fHat)
```

Arguments

fHat	matrix of model-based expected counts

Value

thetaHat vector of association parameters `model_i_effects` *Gets the overall effects for Model I.*

Description

Gets the overall effects for Model I.

Usage

```
model_i_effects(result)
```

Arguments

`result`	a Model I result object

Value

a list containing theta: the overall association zeta_i_dot: row effects for association zeta_dot_j: column effects for association

`model_i_fHat` *Computes model-based expected cell counts for Model I*

Description

Computes model-based expected cell counts for Model I

Usage

```
model_i_fHat(alpha, beta, gamma, delta)
```

Arguments

`alpha`	row effects
`beta`	column effects
`gamma`	row location weights
`delta`	column location weights

Value

matrix of model-based expected counts

`model_i_normalize_fHat`
Normalizes pi(fHat) to sum to 1.0. If exclude_diagonal is TRUE, the sum of the off-diagonal terms sums to 1.0.

Description

Normalizes pi(fHat) to sum to 1.0. If exclude_diagonal is TRUE, the sum of the off-diagonal terms sums to 1.0.

Usage

```
model_i_normalize_fHat(fHat, exclude_diagonal = FALSE)
```

Arguments

`fHat` matrix of model-based cell frequencies

`exclude_diagonal`
logical. Should the cells on the main diagonal be excluded? Default is FALSE, include all cells

Value

matrix of model-based proportions pi

`model_i_row_column_odds_ratios`
Computes the table of adjacent odds-ratios theta-hat.

Description

Computes the table of adjacent odds-ratios theta-hat.

Usage

```
model_i_row_column_odds_ratios(fHat)
```

Arguments

`fHat` matrix of model-based expected counts

Value

thetaHat matrix of adjacent odds-ratios

`model_i_row_theta` *Computes the row association values theta-hat*

Description

Computes the row association values theta-hat

Usage

```
model_i_row_theta(fHat)
```

Arguments

`fHat`	matrix of model-based expected counts

Value

thetaHat vector of association parameters

`model_i_star_effects` *Gets the Model I* effects.*

Description

Gets the Model I* effects.

Usage

```
model_i_star_effects(result)
```

Arguments

`result`	a Model I* effect object

Value

a list containing theta: the overall association zeta: the row/column effect

model_i_star_fHat — *Computes expected frequencies for Model I**

Description

Computes expected frequencies for Model I*

Usage

```
model_i_star_fHat(alpha, beta, theta)
```

Arguments

`alpha`	row effect parameters
`beta`	column effect parameters
`theta`	row/column parameters

Value

matrix of model-based expected cell counts

model_i_star_update_theta — *Updates the row/column parameters for Model I*.*

Description

Updates the row/column parameters for Model I*.

Usage

```
model_i_star_update_theta(theta, n, fHat, exclude_diagonal = FALSE)
```

Arguments

`theta`	vector of estimated row/column effects
`n`	matrix of observed counts
`fHat`	matrix of model-based expected frequencies
`exclude_diagonal`	should the cells of the main diagonal be excluded? Default is FALSE, include all cells

Value

new value of theta vector

model_i_starting_values *Computes crude starting values for Model I.*

Description

Computes crude starting values for Model I.

Usage

```
model_i_starting_values(n)
```

Arguments

n	matrix of observed counts

Value

a list containing alpha: vector of row parameters beta: vector of column parameters gamma: vector of row locations delta: vector of column locations

model_i_update_alpha *Updates the estimate of the alpha vector for Model I*

Description

Updates the estimate of the alpha vector for Model I

Usage

```
model_i_update_alpha(alpha, n, fHat, exclude_diagonal = FALSE)
```

Arguments

alpha	current estimate of beta
n	matrix of observed counts
fHat	current model-based counts for each cell
exclude_diagonal	logical. Should the diagonal be excluded from the computation? Default is FALSE, use all cells.

Value

updated estimate of alpha vector

model_i_update_beta — *Updates the estimate of the beta vector for Model I*

Description

Updates the estimate of the beta vector for Model I

Usage

```
model_i_update_beta(beta, n, fHat, exclude_diagonal = FALSE)
```

Arguments

`beta`	current estimate of alpha
`n`	matrix of observed counts
`fHat`	current model-based counts for each cell
`exclude_diagonal`	logical. Should the diagonal be excluded from the computation? Default is FALSE, use all cells

Value

updated estimate of beta vector

model_i_update_delta — *Updates the estimate of the delta vector for Model I*

Description

Updates the estimate of the delta vector for Model I

Usage

```
model_i_update_delta(delta, n, fHat, exclude_diagonal = FALSE)
```

Arguments

`delta`	current estimate of delta
`n`	matrix of observed counts
`fHat`	current model-based counts for each cell
`exclude_diagonal`	logical. Should the diagonal be excluded from the computation? Default is FALSE, use all cells

Value

updated estimate of delta vector

model_i_update_gamma *Updates the estimate of the gamma vector for Model I*

Description

Updates the estimate of the gamma vector for Model I

Usage

```
model_i_update_gamma(gamma, n, fHat, exclude_diagonal = FALSE)
```

Arguments

gamma	current estimate of gamma
n	matrix of observed counts
fHat	current model-based counts for each cell
exclude_diagonal	logical. Should the diagonal be excluded from the computation? Default is FALSE, use all cells

Value

updated estimate of gamma vector

model_i_zeta *Computes the overall association theta and the row and column effects zeta*

Description

Computes the overall association theta and the row and column effects zeta

Usage

```
model_i_zeta(odds)
```

Arguments

odds	matrix of adjacent odds-ratios

Value

a list containing theta: the overall association zeta_i_dot: row effects for association zeta_dot_j: column effects for association

model_ii_effects *Gets the effects phi, ksi_i_dot and ksi_dot_j for Model II results.*

Description

Gets the effects phi, ksi_i_dot and ksi_dot_j for Model II results.

Usage

```
model_ii_effects(result)
```

Arguments

`result`	a result object from Model II

Value

a list containing: phi: the overall effect ksi_i_dot: the row effects ksi_dot_j: the column effects

model_ii_fHat *Computes expected counts for Model II*

Description

Computes expected counts for Model II

Usage

```
model_ii_fHat(alpha, beta, rho, sigma)
```

Arguments

`alpha`	row effects
`beta`	column effects
`rho`	row locations
`sigma`	column locations

Value

matrix of model-based expected counts

model_ii_ksi — *Gets the effects phi, ksi_i_dot and ksi_dot_j for Model II matrix of odds-ratios.*

Description

Gets the effects phi, ksi_i_dot and ksi_dot_j for Model II matrix of odds-ratios.

Usage

```
model_ii_ksi(odds)
```

Arguments

`odds`	matrix of adjacent odds-ratios

Value

a list containing: phi: the overall effect in log metric ksi_i_dot: the row effects ksi_dot_j: the column effects

model_ii_star_effects — *Gets the effects for Model II**

Description

Gets the effects for Model II*

Usage

```
model_ii_star_effects(result)
```

Arguments

`result`	a Model II* result object

Value

a list containing phi: common effect in log metric ksi: vector of ksi parameters

model_ii_star_fHat *Computes expected counts for Model II**

Description

Computes expected counts for Model II*

Usage

```
model_ii_star_fHat(alpha, beta, phi)
```

Arguments

alpha	row effects
beta	column effects
phi	row/column locations

Value

matrix of model-based expected counts

model_ii_star_update_phi *Updates estimate of phi vector*

Description

Updates estimate of phi vector

Usage

```
model_ii_star_update_phi(n, fHat, mu, phi, exclude_diagonal = FALSE)
```

Arguments

n	matrix of observed counts
fHat	current model-based counts for each cell
mu	alternative row coefficients
phi	vector of column location parameters
exclude_diagonal	logical, Should the cells on the main diagonal be excluded? Default is FALSE, use all cells

Value

list containing: phi: updated estimate of the phi vector mu: updated estimate of vector mu

model_ii_starting_values *Computes crude starting values for Model II*

Description

Computes crude starting values for Model II

Usage

```
model_ii_starting_values(n)
```

Arguments

`n`	matrix of observed counts

Value

a list containing alpha: vector of row parameters beta: vector of column parameters rho: row coefficients sigma: column coefficients mu: alternative row coefficients nu: alternative column coefficients

model_ii_update_alpha *Updates the estimate of the alpha vector for Model II*

Description

Updates the estimate of the alpha vector for Model II

Usage

```
model_ii_update_alpha(alpha, n, fHat, exclude_diagonal = FALSE)
```

Arguments

`alpha`	current estimate of alpha
`n`	matrix of observed counts
`fHat`	current model-based counts for each cell
`exclude_diagonal`	logical, Should the cells on the main diagonal be excluded? Default is FALSE, use all cells

Value

updated estimate of alpha vector

model_ii_update_beta *Updates the estimate of the beta vector for Model II*

Description

Updates the estimate of the beta vector for Model II

Usage

```
model_ii_update_beta(beta, n, fHat, exclude_diagonal = FALSE)
```

Arguments

`beta`	current estimate of beta
`n`	matrix of observed counts
`fHat`	current model-based counts for each cell
`exclude_diagonal`	logical, Should the cells on the main diagonal be excluded? Default is FALSE, use all cells

Value

updated estimate of beta vector

model_ii_update_rho *Updates the estimate of the rho vector for Model II*

Description

Updates the estimate of the rho vector for Model II

Usage

```
model_ii_update_rho(n, fHat, mu, sigma, exclude_diagonal = FALSE)
```

Arguments

`n`	matrix of observed counts
`fHat`	current model-based counts for each cell
`mu`	alternative row coefficients
`sigma`	vector of column location parameters
`exclude_diagonal`	logical, Should the cells on the main diagonal be excluded? Default is FALSE, use all cells

Value

updated estimate of alpha vector

---

model_ii_update_sigma *Updates the estimate of the sigma vector for Model II*

Description

Updates the estimate of the sigma vector for Model II

Usage

```
model_ii_update_sigma(n, fHat, nu, rho, exclude_diagonal = FALSE)
```

Arguments

n	matrix of observed counts
fHat	current model-based counts for each cell
nu	vector of column coefficients
rho	vector of row location parameters
exclude_diagonal	logical, Should the cells on the main diagonal be excluded? Default is FALSE, use all cells

Value

updated estimate of sigma vector

---

movies *Movie ratings by two film critics, Siskel and Ebert.*

Description

Movie ratings by two film critics, Siskel and Ebert.

Usage

```
movies
```

Format

‘movies‘ A matrix with 3 rows and 3 columns 1 is con 2 is mixed 3 is pro

Source

https://online.stat.psu.edu/stat504/lesson/11/11.3

new_orleans_data — *Agreement between two clinicians on presence of multiple sclerosis based on file.*

Description

See companion winnipeg_data.

Usage

```
new_orleans_data
```

Format

'new_orleans_data' A matrix with 4 rows and 4 columns Ratings range from definite presence of disease to definite absence.

Source

???

null_association_fHat — *Computes expected counts for null association model*

Description

Computes expected counts for null association model

Usage

```
null_association_fHat(alpha, beta)
```

Arguments

alpha	row effects
beta	column effects

Value

matrix of model-based expected counts

`occupational_status` *Cross tabulation of father's employment status with son's employment status.*

Description

Higher numbers correspond to higher status occupation

Usage

```
occupational_status
```

Format

'occupational_status' A matrix with 6 rows and 6 columns

Source

???

`paranoia` *Interrater agreement of two psychologists' ratings of paranoia.*

Description

Severity corresponds to level 1 low 3 high

Usage

```
paranoia
```

Format

'paranoia' A matrix with 3 rows and 3 columns.

Source

von Eye, A. & Mun, E. Y. (2005, p. 70). Analyzing rater agreement: Manifest variable methods. Mahwah, NJ: Lawrence Erlbaum.

`pearson_chisq` *Computes the Pearson X^2 statistic.*

Description

Computes the Pearson X^2 statistic.

Usage

```
pearson_chisq(n, pi, exclude_diagonal = FALSE)
```

Arguments

`n`	Matrix of observed counts
`pi`	Matrix with same dimensions as n. Model-based matrix of predicted proportions
`exclude_diagonal`	logical. Should diagonal cells of square matrix be excluded from the computation? Default is FALSE. The effect of setting it to TRUE for non-square matrices may be unintuitive and should he avoided.

Value

X^2

`radiology` *Interrater agreement of two radiologists diagnosis of severity of carcinoma.*

Description

The data contains a comparison vector of (simulated) covariate data.

Usage

```
radiology
```

Format

‘radiology‘ ‘covariate‘ A matrix with 4 rows and 4 columns, and a vector of 16 elements.

Source

von Eye, A. & Mun, E. Y. (2005, p. 60). Analyzing rater agreement: Manifest variable methods. Mahwah, NJ: Lawrence Erlbaum.

Schuster_compute_df *Computes the degrees of freedom for the model.*

Description

Computes the degrees of freedom for the model.

Usage

```
Schuster_compute_df(pi_margin)
```

Arguments

pi_margin	expected proportions for each of the categories

Value

the df for the model

Schuster_compute_pi *Compute matrix of model-based proportions pi.*

Description

Compute matrix of model-based proportions pi.

Usage

```
Schuster_compute_pi(marginal_pi, kappa, v, validate = TRUE)
```

Arguments

marginal_pi	expected proportions for each category
kappa	current estimate of the kappa coefficient
v	symmetry matrix
validate	logical. should the cells be validated within this function? Defaults to TRUE

Value

matrix of model-based cell proportions

`Schuster_compute_starting_values`
Computes starting values for the model.

Description

Patterned after example in code in appendix to article

Usage

```
Schuster_compute_starting_values(n)
```

Arguments

`n`	matrix of observed counts

Value

a list containing marginal_pi: vector of expected proportions for each category kappa: kappa coefficient of agreement v: matrix of symmetry parameters

---

`Schuster_derivative_log_l_wrt_kappa`
Derivative of log(likelihood) wrt kappa.

Description

Derivative of log(likelihood) wrt kappa.

Usage

```
Schuster_derivative_log_l_wrt_kappa(n, marginal_pi, kappa, v)
```

Arguments

`n`	matrix of observed counts
`marginal_pi`	expected proportions for each category
`kappa`	current value of kappa coefficient
`v`	symmetry matrix

Value

derivative of log(L) wrt kappa

Schuster_derivative_log_l_wrt_marginal_pi

Derivative of log(likelihood) wrt marginal_pi[k]

Description

Derivative of log(likelihood) wrt marginal_pi[k]

Usage

```
Schuster_derivative_log_l_wrt_marginal_pi(n, k, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
k	index into marginal_pi
marginal_pi	expected proportions of each of the categories
kappa	current value of the kappa coefficient
v	symmetry matrix

Value

derivative of log(L) wrt marginal_pi[k]

Schuster_derivative_log_l_wrt_v

Derivative of log(likelihood) wrt v[i1, j1]

Description

Derivative of log(likelihood) wrt v[i1, j1]

Usage

```
Schuster_derivative_log_l_wrt_v(n, i1, j1, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
i1	first index into v
j1	second index into v
marginal_pi	expected marginal proportions
kappa	current value of kappa coefficient
v	symmetry matrix

Value

derivative of log(L) wrt v[i1, j1]

Schuster_derivative_pi_wrt_kappa

Derivative of pi[i, j] wrt kappa coefficient.

Description

Derivative of pi[i, j] wrt kappa coefficient.

Usage

```
Schuster_derivative_pi_wrt_kappa(i, j, marginal_pi, kappa, v)
```

Arguments

i	first index into pi
j	second index into pi
marginal_pi	expected proportions in each category
kappa	current value of kappa coefficient
v	symmetry matrix

Value

the derivative of pi[i, j] wrt kappa

Schuster_derivative_pi_wrt_marginal_pi

Derivative of pi[i, j] wrt marginal_pi[k].

Description

Derivative of pi[i, j] wrt marginal_pi[k].

Usage

```
Schuster_derivative_pi_wrt_marginal_pi(i, j, k, marginal_pi, kappa, v)
```

Arguments

`i`	first index into pi
`j`	second index into pi
`k`	index into marginal_pi
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

derivative of pi[i, j] wrt marginal_pi[k]

`Schuster_derivative_pi_wrt_v` *Computes derivative of pi[i, j] wrt v[i1, j1]*

Description

Computes derivative of pi[i, j] wrt v[i1, j1]

Usage

```
Schuster_derivative_pi_wrt_v(i, j, i1, j1, marginal_pi, kappa, v)
```

Arguments

`i`	first index into pi
`j`	second index into pi
`i1`	first index into v
`j1`	second index into v
`marginal_pi`	expected marginal proportions
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

value of derivative of specified pi wrt specified element of v

Schuster_derivative_v_wrt_v

Computes derivative of v[i1, j1] wrt v[i2, j2]

Description

Needed because of computed v terms in column r

Usage

```
Schuster_derivative_v_wrt_v(i1, j1, i2, j2, marginal_pi, kappa, v)
```

Arguments

i1	first index into target v
j1	second index into target v
i2	first index into
j2	second index into
marginal_pi	expected marginal proportions
kappa	current estimate of kappa coefficient
v	matrix of symmetry parameters

Value

derivative of v[i1, j1] wrt v[i2, j2]

Schuster_enforce_constraints_on_v

Compute v matrix subject to constraints on rows 1..r-1.

Description

Compute v matrix subject to constraints on rows 1..r-1.

Usage

```
Schuster_enforce_constraints_on_v(marginal_pi, kappa, v)
```

Arguments

marginal_pi	expected proportions for each category
kappa	current estimate of kappa coefficient
v	symmetry matrix

Value

new v matrix with last row/column set to agree with constraints. Element v[r, r] is set to v-tilde

---

`Schuster_gradient` *Gradient vector log(L) wrt parameters.*

Description

Work is delegated to functions that compute partial derivatives. This function is responsible for laying them out in correct positions in the vector.

Usage

```
Schuster_gradient(n, marginal_pi, kappa, v)
```

Arguments

`n`	matrix of observed counts
`marginal_pi`	expected proportions for each response category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

gradient vector

`Schuster_hessian` *Computes the hessian matrix of second-order partial derivatives of log(L).*

Description

Work is delegated to functions that compute second-order partial derivatives. This function is responsible for laying them out in correct positions in the matrix.

Usage

```
Schuster_hessian(n, marginal_pi, kappa, v)
```

Arguments

`n`	matrix of observed counts
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of the kappa coefficient
`v`	symmetry matrix

Value

hessian matrix

`Schuster_is_pi_valid` *Determines whether the candidate pi matrix is valid.*

Description

All elements must lie in (0, 1)

Usage

```
Schuster_is_pi_valid(pi)
```

Arguments

`pi`	matrix of model-based proportions

Value

logical value indicating whether or not the matrix is valid.

`Schuster_newton_raphson`
Performs Newton-Raphson step.

Description

The step size is determined to be the largest that yields valid results for all quantities marginal_pi and v. Both must be positive, and the elements of marginal_pi must be valid proportions that sum to 1.0.

Usage

```
Schuster_newton_raphson(n, marginal_pi, kappa, v)
```

Arguments

`n`	matrix of observed counts
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of the kappa coefficient
`v`	symmetry matrix

Value

a list containing updated versions of model quantities marginal_pi kappa v

`Schuster_second_deriv_log_l_wrt_kappa_2`
Second order partial log(L) wrt kappa^2.

Description

Second order partial log(L) wrt kappa^2.

Usage

```
Schuster_second_deriv_log_l_wrt_kappa_2(n, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
marginal_pi	expected proportions for each response category
kappa	current estimate of kappa coefficient
v	symmetry matrix second derivative of log(L) wrt kappa^2

`Schuster_second_deriv_log_l_wrt_kappa_v`
Second order partial log(L) wrt kappa and v.

Description

Second order partial log(L) wrt kappa and v.

Usage

```
Schuster_second_deriv_log_l_wrt_kappa_v(n, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
marginal_pi	expected proportions for each response category
kappa	current estimate of kappa coefficient
v	symmetry matrix second derivative of log(L) wrt kappa and v

Schuster_second_deriv_log_l_wrt_marginal_pi_2

Second order partial log(L) wrt marginal_pi^2.

Description

Second order partial log(L) wrt marginal_pi^2.

Usage

```
Schuster_second_deriv_log_l_wrt_marginal_pi_2(n, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
marginal_pi	expected proportions for each response category
kappa	current estimate of kappa coefficient
v	symmetry matrix second derivative of log(L) wrt marginal_pi^2

Schuster_second_deriv_log_l_wrt_marginal_pi_kappa

Second order partial log(L) wrt marginal_pi and kappa.

Description

Second order partial log(L) wrt marginal_pi and kappa.

Usage

```
Schuster_second_deriv_log_l_wrt_marginal_pi_kappa(n, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
marginal_pi	expected proportions for each response category
kappa	current estimate of kappa coefficient
v	symmetry matrix second derivative of log(L) wrt marginal_pi and kappa

Schuster_second_deriv_log_l_wrt_marginal_pi_v

Second order partial log(L) wrt marginal_pi and v.

Description

Second order partial log(L) wrt marginal_pi and v.

Usage

```
Schuster_second_deriv_log_l_wrt_marginal_pi_v(n, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
marginal_pi	expected proportions for each response category
kappa	current estimate of kappa coefficient
v	symmetry matrix second derivative of log(L) wrt marginal_pi and v

Schuster_second_deriv_log_l_wrt_v_2

Second order partial log(L) wrt v^2.

Description

Second order partial log(L) wrt v^2.

Usage

```
Schuster_second_deriv_log_l_wrt_v_2(n, marginal_pi, kappa, v)
```

Arguments

n	matrix of observed counts
marginal_pi	expected proportions for each response category
kappa	current estimate of kappa coefficient
v	symmetry matrix second derivative of log(L) wrt v^2

Schuster_second_deriv_pi_wrt_kappa_2

Second order partial wrt kappa, kappa

Description

Derivative is uniformly 0

Usage

```
Schuster_second_deriv_pi_wrt_kappa_2(i, j, marginal_pi, kappa, v)
```

Arguments

`i`	first index of pi
`j`	second index of pi
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of the kappa coefficient
`v`	symmetry matrix

Value

second order partial derivative

Schuster_second_deriv_pi_wrt_kappa_v

Second order partial wrt kappa, v

Description

Derivative is uniformly 0

Usage

```
Schuster_second_deriv_pi_wrt_kappa_v(i, j, i1, j1, marginal_pi, kappa, v)
```

Arguments

`i`	first index of pi
`j`	second index of pi
`i1`	first index of v
`j1`	second index of v
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of the kappa coefficient
`v`	symmetry matrix

Value

second order partial derivative

Schuster_second_deriv_pi_wrt_marginal_pi_2

Second derivative of pi[i, j] wrt marginal_pi[k]^2

Description

Second derivative of pi[i, j] wrt marginal_pi[k]^2

Usage

```
Schuster_second_deriv_pi_wrt_marginal_pi_2(i, j, k, k2, marginal_pi, kappa, v)
```

Arguments

i	first index into pi
j	second index into pi
k	index into marginal_pi
k2	second index into marginal_pi
marginal_pi	expected proportions for each category
kappa	current estimate of kappa coefficient
v	symmetry matrix

Value

second derivative of pi[i, j] wrt marginal_pi^2

Schuster_second_deriv_pi_wrt_marginal_pi_kappa

Second order partial wrt kappa, marginal_pi

Description

Derivative is uniformly 0

Usage

```
Schuster_second_deriv_pi_wrt_marginal_pi_kappa(i, j, k, marginal_pi, kappa, v)
```

Arguments

`i`	first index of pi
`j`	second index of pi
`k`	index of marginal_pi
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of the kappa coefficient
`v`	symmetry matrix

Value

second order partial derivative

`Schuster_second_deriv_pi_wrt_marginal_pi_v` *Second order partial pi wrt marginal_pi and v*

Description

Second order partial pi wrt marginal_pi and v

Usage

```
Schuster_second_deriv_pi_wrt_marginal_pi_v(
  i,
  j,
  k,
  i1,
  j1,
  marginal_pi,
  kappa,
  v
)
```

Arguments

`i`	first index of pi
`j`	second index of pi
`k`	index of marginal_pi
`i1`	first index of v
`j1`	second index of v
`marginal_pi`	expected proportions of each of the categories
`kappa`	current value of kappa coefficient
`v`	symmetry matrix

Value

derivative

`Schuster_second_deriv_pi_wrt_v_2`
Second order partial wrt v^2

Description

Derivative is uniformly 0

Usage

```
Schuster_second_deriv_pi_wrt_v_2(i, j, i1, j1, i2, j2, marginal_pi, kappa, v)
```

Arguments

`i`	first index of pi
`j`	second index of pi
`i1`	first index of first v
`j1`	second index of first v
`i2`	first index of second v
`j2`	second index of second
`marginal_pi`	expected proportions for each category
`kappa`	current estimate of the kappa coefficient
`v`	symmetry matrix

Value

second order partial derivative

`Schuster_solve_for_v` *Solves for the last row and diagonal of symmetry matrix v (v-tilde) using constraint equations*

Description

Solves for the last row and diagonal of symmetry matrix v (v-tilde) using constraint equations

Usage

```
Schuster_solve_for_v(marginal_pi, kappa, v)
```

Arguments

`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

revised version of v matrix with last row and diagonal modified

`Schuster_solve_for_v1` *Solves for the last row and diagonal of symmetry matrix v (parameteer v-tilde) using linear algebra formulation from paper.*

Description

Solves for the last row and diagonal of symmetry matrix v (parameteer v-tilde) using linear algebra formulation from paper.

Usage

```
Schuster_solve_for_v1(marginal_pi, kappa, v)
```

Arguments

`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`v`	symmetry matrix

Value

revised version of v matrix with last row and diagonal modified

Schuster_symmetric_rater_agreement_model

Computes the model that has kappa as a coefficient and symmetry.

Description

Schuster, C. (2001). Kappa as a parameter of a symmetry model for rater agreement. Journal of Educational and Behavioral Statistics, 26(3), 331-342.

Usage

```
Schuster_symmetric_rater_agreement_model(
  n,
  verbose = FALSE,
  max_iter = 10000,
  criterion = 1e-07,
  min_iter = 1000
)
```

Arguments

n	the matrix of observed counts
verbose	logical. should cycle-by-cycle information be printed out
max_iter	integer. maximum number of iterations to perform
criterion	number. maximum change in log(likelihood) to decide convergence
min_iter	integer. minimum number of iterations to perform

Value

a list containing marginal_pi: vector of expected proportions for each category kappa numeric: kappa coefficient v: matrix of symmetry parameters chisq: Pearson X^2 g_squared: likelihood ratio G^2 df: degrees of freedom

Schuster_update

Computes the Newton-Raphson update

Description

Computes both gradient and hessian, and then solves the system of equations

Usage

```
Schuster_update(n, marginal_pi, kappa, v)
```

Arguments

`n`	matrix of observed counts
`marginal_pi`	expected proportions for each category
`kappa`	current value of kappa coefficient
`v`	symmetry matrix

Value

the vector of updates

`Schuster_v_tilde` *Computes the common diagonal term v-tilde.*

Description

Computes the common diagonal term v-tilde.

Usage

```
Schuster_v_tilde(marginal_pi, kappa, validate = TRUE)
```

Arguments

`marginal_pi`	expected proportions for each category
`kappa`	current estimate of kappa coefficient
`validate`	logical. should the value of pi[r,r] be checked for validity? Default is TRUE

Value

v-tilde

`social_status` *Social mobility data with father's occupational social status and son's occupational social status.*

Description

Social mobility data with father's occupational social status and son's occupational social status.

Usage

```
social_status
```

Format

‘social_status‘ A matrix with 7 rows and 7 columns

Source

Goodman, L. A. (1979). Simple models for the analysis of association in cross-classifications having ordered categories. Journal of the American Statistical Association, 74(367), 537-552.

`social_status2` *Social mobility data with father's occupational social status and son's occupational social status. * categories instead of 7 in social status..*

Description

Social mobility data with father's occupational social status and son's occupational social status. * categories instead of 7 in social status..

Usage

```
social_status2
```

Format

‘social_status2‘ A matrix with 8 rows and 8 columns

Source

Goodman, L. A. (1979). Simple models for the analysis of association in cross-classifications having ordered categories. Journal of the American Statistical Association, 74(367), 537-552.

`Stuart_marginal_homogeneity`
Computes Stuart's Q test of marginal homogeneity.

Description

Stuart, A. (1955). A test for homogeneity of the marginal distributions in a two-way classification. Biometrika, 42(3/4), 412-416.

Usage

```
Stuart_marginal_homogeneity(n)
```

Arguments

`n`	matrix of observed counts

Value

a list containing q: value of q test-statistic df: degrees of freedom p: upper tail p-value of q

Examples

```
Stuart_marginal_homogeneity(vision_data)
```

---

taste *Taste ratings*

Description

Taste ratings

Usage

```
taste
```

Format

'taste' A matrix with 5 rows and 5 columns.

Source

McCullagh, P. (1980, p. 119). Regression models for ordinal data. Journal of the Royal Statistical Society, Series B, 42(2), 109-142.

---

teachers *Teachers ratings of their students intelligence.*

Description

Interrater agreement data for two teachers asked to rate the intelligence of their students.

Usage

```
teachers
```

Format

'teachers' A matrix with 4 rows and 4 columns. Higher scores correspond to higher estimated intelligence.

Source

von Eye, A. & Mun, E. Y. (2005, p. 36). Analyzing rater agreement: Manifest variable methods. Mahwah, NJ: Lawrence Erlbaum.

teaching_style — *Style of teachers rated by supervisors*

Description

Ratings of style of teaching by supervisors. 1 indicates Authoritarian, 2 indicates Democratic, 3 indicates Permissive.

Usage

```
teaching_style
```

Format

An object of class `matrix` (inherits from `array`) with 3 rows and 3 columns.

Details

@format ## ‘teaching_style‘ A matrix with 3 rows and 3 columns.

@source Agresti, A. (1989). An agreement model with kappa as parameter. Statistics & Probability Letters, 7, 271-273.

tonsils — *Relationship between size of child’s tonsils and their status as a carrier of a disease.*

Description

Relationship between size of child’s tonsils and their status as a carrier of a disease.

Usage

```
tonsils
```

Format

‘tonsils‘ A matrix with 2 rows and 3 columns. Rows are disease status and columns are ratings of tonsil size.

Source

McCullagh, P. (1980). Regression models for ordinal data. Journal of the Royal Statistical Society, Series B, 42(2), 109-142.

tv — *Interrater agreement of two journalists' evaluation of proposed TV programs.*

Description

Ratings go from low to high probability of the show's success.

Usage

```
tv
```

Format

'tv' A matrix of 6 rows and 6 columns.

Source

von Eye, A. & Mun, E. Y. (2005, p. 56). Analyzing rater agreement: Manifest variable methods. Mahwah, NJ: Lawrence Erlbaum.

uniform_association_fHat — *Computes expected counts for uniform association model*

Description

Computes expected counts for uniform association model

Usage

```
uniform_association_fHat(alpha, beta, theta)
```

Arguments

`alpha`	row effects
`beta`	column effects
`theta`	association parameter

Value

matrix of model-based expected counts

uniform_association_update_theta

Updates estimate of theta value of the uniform association model

Description

Updates estimate of theta value of the uniform association model

Usage

```
uniform_association_update_theta(theta, n, fHat, exclude_diagonal = FALSE)
```

Arguments

theta	current estimate of theta
n	matrix of observed counts
fHat	current model-based counts for each cell
exclude_diagonal	logical. Should the cells of the main diagonal be excluded from the computations? Defualt is FALSE, include all cells.

Value

updated estimate of theta parameter

var_kappa

Computes the sampling variance of kappa.

Description

Formulas are from the paper by Fleiss,J. L., Cohen, J., & Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. Two results are returned in a list. var_kappa0 is the null case and would be used for testing the hypothesis that kappa = 0. The second is var_kappa and is for the non-null case, such as constructing CI for estimated kappa. Not that both are in the variance metric. Take the square root to get the standard error.

Usage

```
var_kappa(n)
```

Arguments

n	matrix of observe counts

Value

a list containing; var_kappa0: variance for the null case var_kappa: variance for the non-null case.

var_weighted_kappa — *Computes the sampling variance of weighted kappa.*

Description

Formulas are from the paper by Fleiss,J. L., Cohen, J., & Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. Two results are returned in a list. var_kappa0 is the null case and would be used for testing the hypothesis that kappa = 0. The second is var_kappa and is for the non-null case, such as constructing CI for estimated kappa. Not that both are in the variance metric. Take the square root to get the standard error.

Usage

```
var_weighted_kappa(n, w)
```

Arguments

n	matrix of observe counts
w	matrix of penalty weights

Value

a list containing; var_kappa0: variance for the null case var_kappa: variance for the non-null case.

vision_data — *Visual acuity of women factory workers.*

Description

Measurements of unaided visual acuity for women working at the Royal Ordinance factories 1943-1946. Rows are right eye, columns are left eye. 1 indicates best vision, 4 is poorest.

Usage

```
vision_data
```

Format

‘visual_data‘ A matrix with 4 rows and 4 columns.

Source

Stuart, A. (1953). The estimation and comparison of strengths of association in contingency tables. Biometrika, 40(1/2), 105-110.

vision_data_men *Visual acuity of men factory workers.*

Description

Measurements of unaided visual acuity for men working at the Royal Ordinance factories 1943-1946. Rows are right eye, columns are left eye. 1 indicates best vision, 4 is poorest.

Usage

```
vision_data_men
```

Format

‘visual_data_men‘ A matrix with 4 rows and 4 columns.

Source

Stuart, A. (1953). The estimation and comparison of strengths of association in contingency tables. Biometrika, 40(1/2), 105-110.

von_Eye_diagonal *Fits the diagonal effects model, where each category has its own parameter delta[k].*

Description

Fits the diagonal effects model, where each category has its own parameter delta[k].

Usage

```
von_Eye_diagonal(n)
```

Arguments

n	the matrix of observed counts

Value

a list containing beta: the regression parameters. delta parameters are the final elements of beta g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

von_Eye_diagonal_linear_by_linear

Fits the diagonal effects model, where each category has its own parameter delta[k], while also incorporating a linear-by-linear term.

Description

Fits the diagonal effects model, where each category has its own parameter delta[k], while also incorporating a linear-by-linear term.

Usage

```
von_Eye_diagonal_linear_by_linear(n, center = TRUE)
```

Arguments

`n`	the matrix of observed counts
`center`	should the linear-by-linear components be centered to have mean 0? Default is TRUE

Value

a list containing beta: the regression parameters. delta parameters come after rows and columns and finally the linear-by-linear term g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

von_Eye_equal_weight_diagonal_linear

Fits the diagonal effects model, where there is a single delta parameter for all categories, while also incorporating a linear-by-linear term.

Description

Fits the diagonal effects model, where there is a single delta parameter for all categories, while also incorporating a linear-by-linear term.

Usage

```
von_Eye_equal_weight_diagonal_linear(n, center = TRUE)
```

Arguments

`n`	the matrix of observed counts
`center`	should the linear-by-linear components be centered to have mean 0? Default is TRUE

Value

a list containing beta: the regression parameters. delta parameters come after rows and columns and finally the linear-by-linear term g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

---

von_Eye_equal_weighted_diagonal

Fits the equal weighted diagonal model, where the diagonals all have an additional parameter delta, with the constraint that delta is equal across all categories.

Description

Fits the equal weighted diagonal model, where the diagonals all have an additional parameter delta, with the constraint that delta is equal across all categories.

Usage

```
von_Eye_equal_weighted_diagonal(n)
```

Arguments

`n`	the matrix of observed counts

Value

a list containing beta: the regression parameters g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

---

von_Eye_linear_by_linear

Fits the basic independent rows and columns model incorporating a linear-by-linear term.

Description

Fits the basic independent rows and columns model incorporating a linear-by-linear term.

Usage

```
von_Eye_linear_by_linear(n, center = TRUE)
```

Arguments

`n` matrix of observed counts

`center` should the linear-by-linear components be centered to have mean 0? Default is TRUE

Value

a list containing beta: the regression parameters. The linear-by-linear parameter is last g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

---

von_Eye_main_effect *Fits the base model with only independent row and column effects.*

Description

Fits the base model with only independent row and column effects.

Usage

```
von_Eye_main_effect(n)
```

Arguments

`n` the matrix of observed counts

Value

a list containing beta: the regression parameters g_squared: G^2 fit measure chisq: X^2 fit measure df: degrees of freedom expected: matrix of expected frequencies

---

von_Eye_weight_by_response_category_design *Creates design matrix for weight be response category model.*

Description

The model specifies main effects for row and column, and a parameter for the agreement (diagonal) cells. This takes a design matrix for that model and applies domain-specific weights to the agreement parameters.

Usage

```
von_Eye_weight_by_response_category_design(n, x, w, n_raters = 2)
```

Arguments

`n`	the matrix of cell counts
`x`	the original design matrix.
`w`	the vector of weights to apply to the agreement cells. Should have same number of entries as the number of diagonal elements (number of rows & of columns)
`n_raters`	number of raters. Currently only 2 (the default) are supported. This is an extension point for future work.

Value

new design matrix with weights applied to the agreement cells.

`weighted_cov` *Computes the weighted covariance*

Description

Computes covariance between x and y using case weights in w

Usage

```
weighted_cov(x, y, w, use_df = TRUE)
```

Arguments

`x`	Numeric vector. First variable
`y`	Numeric vector. Second variable
`w`	Numeric vector. case weights
`use_df`	Logical. should the divisor be sum of weights - 1 (TRUE) or N - 1 (FALSE)

Value

the weighted covariance between x and y

weighted_kappa — *Computes Cohen's 1968 weighted kappa coefficient*

Description

Computes Cohen's 1968 weighted kappa coefficient

Usage

```
weighted_kappa(n, w = diag(rep(1, nrow(n))), quadratic = FALSE)
```

Arguments

`n`	matrix of observed counts
`w`	matrix of weights. Defaults to identity matrix
`quadratic`	logical. Should quadratic weights be used? Default is FALSE. If TRUE, quadratic weights are used. These override the values in w. If FALSE, weights in w are used

Value

value of weighted kappa

weighted_var — *Computes the weighted variance*

Description

Computes variance between x and y using case weights in w

Usage

```
weighted_var(x, w, use_df = TRUE)
```

Arguments

`x`	Numeric vector. First variable
`w`	Numeric vector. Case weights
`use_df`	Logical. Should the divisor be sum of weights - 1 (TRUE) or N - 1 (FALSE)

Value

the weighted covariance between x and y

`winnipeg_data` *Agreement between two clinicians on presence of multiple sclerosis based on file.*

Description

See companion new_orleans_data.

Usage

```
winnipeg_data
```

Format

'winnipeg_data' A matrix with 4 rows and 4 columns Ratings range from definite presence of disease to definite absence.

Source

???

Index