Package 'MRCV'

September 4, 2014

Version 0.3-3

Date 2014-09-03

Title Methods for Analyzing Multiple Response Categorical Variables (MRCVs)

Author Natalie Koziol and Chris Bilder

Maintainer Natalie Koziol <nak371@gmail.com>

Depends R (>= 3.1.1)

Imports tables

Suggests geepack

LazyData TRUE

Description The MRCV package provides functions for analyzing the association between one single response categorical variable (SRCV) and one multiple response categorical variable (MRCV), or between two or three MRCVs. A modified Pearson chi-square statistic can be used to test for marginal independence for the one or two MRCV case, or a more general loglinear modeling approach can be used to examine various other structures of association for the two or three MRCV case. Bootstrap- and asymptotic-based standardized residuals and model-predicted odds ratios are available, in addition to other descriptive information.

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-04 06:55:56

R topics documented:

`MRCV-package` *Methods for Analyzing Multiple Response Categorical Variables*

Description

The MRCV package provides functions for analyzing the association between one single response categorical variable (SRCV) and one multiple response categorical variable (MRCV), or between two or three MRCVs. A modified Pearson chi-square statistic can be used to test for marginal independence for the one or two MRCV case, or a more general loglinear modeling approach can be used to examine various other structures of association for the two or three MRCV case. Bootstrap- and asymptotic-based standardized residuals and model-predicted odds ratios are available, in addition to other descriptive information.

Details

Package:	MRCV
Version:	0.3-3
Date:	2014-09-03
Depends:	R (>= 3.1.1)
Imports:	tables
Suggests:	geepack
LazyData:	TRUE
License:	GPL (>= 3)

Notation:

For the two or three MRCV case, define row variable, W, column variable, Y, and strata variable, Z, as MRCVs with binary items (i.e., categories) Wi for i = 1, . . . , I, Yj for j = 1, . . . , J, and Zk for k = 1, . . . , K, respectively. Define a marginal count as the number of subjects who responded (Wi = a, Yj = b, Zk = c) for a, b, and c belonging to the set {0, 1}. For the one MRCV case, let W be an SRCV such that I = 1 and 'a' corresponds to one of r levels of W, and let Y be the MRCV as previously defined.

Format of Data Frame:
Many of the functions require a data frame containing the raw data structured such that the *n* rows correspond to the individual item response vectors, and the columns correspond to the items, W1, . . . , WI, Y1, . . . , YJ, and Z1, . . . , ZK (in this order). Some of the functions use a summary version of the raw data frame (converted automatically without need for user action) formatted to have rx2J rows and 4 columns generically named `W`, `Y`, `yj`, and `count` (one MRCV case), 2Ix2J rows and 5 columns named `W`, `Y`, `wi`, `yj`, and `count` (two MRCV case), or 2Ix2Jx2K rows and 7 columns named `W`, `Y`, `Z`, `wi`, `yj`, `zk`, and `count` (three MRCV case). The column named `count` contains the marginal counts defined above.

Descriptive Functions:
Users can call the `item.response.table` function to obtain a cross-tabulation of the positive and negative responses for each combination of items, or the `marginal.table` function to obtain a cross-tabulation of only the positive responses.

Functions to Test for Marginal Independence:
Methods proposed by Agresti and Liu (1999), Bilder and Loughin (2004), Bilder, Loughin, and Nettleton (2000), and Thomas and Decady (2004) are implemented using the `MI.test` function. This function calculates a modified Pearson chi-square statistic that can be used to test for multiple marginal independence (MMI; one MRCV case) or simultaneous pairwise marginal independence (SPMI; two MRCV case). MMI is a test of whether the SRCV, W, is marginally independent of each Yj, where the modified statistic is the sum of the J Pearson statistics used to test for independence of each (W, Yj) pair. SPMI is a test of whether each Wi is pairwise independent of each Yj, where the modified statistic is the sum of the IxJ Pearson statistics used to test for independence of each (Wi, Yj) pair. The asymptotic distribution of the modified statistics is a linear combination of independent chi-square(1) random variables, so traditional methods for analyzing the association between categorical variables W and Y are inappropriate. The `MI.test` function offers three sets of testing methods: a nonparametric bootstrap approach, a Rao-Scott second-order adjustment, and a Bonferroni adjustment, that can be used in conjunction with the modified statistic to construct an appropriate test for independence.

Functions for Performing Regression Modeling:
Regression modeling methods described by Bilder and Loughin (2007) are implemented using `genloglin` and methods `summary.genloglin`, `residuals.genloglin`, `anova.genloglin`, and `predict.genloglin`. The `genloglin` function provides parameter estimates and Rao-Scott adjusted standard errors for models involving two or three MRCVs. The `anova.genloglin` function offers second-order Rao-Scott and bootstrap adjusted model comparison and goodness-of-fit (Pearson and LRT) statistics. The `residuals.genloglin` and `predict.genloglin` functions provide bootstrap- and asymptotic-based standardized Pearson residuals and model-based odds ratios, respectively.

General Notes:
Rao-Scott adjustments may not be feasible when the total number of MRCV items is large. In this case, an error message will be returned describing a memory allocation issue.

Author(s)

Natalie Koziol, Chris Bilder

Maintainer: Natalie Koziol <nak371@gmail.com>

References

Agresti, A. and Liu, I.-M. (1999) Modeling a categorical variable allowing arbitrarily many category choices. *Biometrics*, **55**, 936–943.

Bilder, C. and Loughin, T. (2004) Testing for marginal independence between two categorical variables with multiple responses. *Biometrics*, **36**, 433–451.

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

Bilder, C., Loughin, T., and Nettleton, D. (2000) Multiple marginal independence testing for pick any/c variables. *Communications in Statistics–Theory and Methods*, **29**, 1285–1316.

Thomas, D. and Decady, Y. (2004) Testing for association using multiple response survey data: Approximate procedures based on the Rao-Scott approach. *International Journal of Testing*, **4**, 43–59.

`anova.genloglin` *Perform MRCV Model Comparison Tests*

Description

The `anova.genloglin` method function offers second-order Rao-Scott and bootstrap adjusted model comparison and goodness-of-fit (Pearson and LRT) statistics appropriate for evaluating models estimated by the `genloglin` function.

Usage

```
## S3 method for class 'genloglin'
anova(object, model.HA = "saturated", type = "all", gof = TRUE,
    print.status = TRUE, ...)
```

Arguments

`object`	An object of class `'genloglin'` produced by the `genloglin` function.
`model.HA`	For the two MRCV case, a character string specifying one of the following models to be compared to the null model (where the null model should be nested within the alternative model): `"homogeneous"` (the homogeneous association model), `"w.main"` (the w-main effects model), `"y.main"` (the y-main effects model), `"wy.main"` (the w- and y-main effects model), or `"saturated"`. Alternatively, a user-supplied formula can be specified. For the three MRCV case, only `"saturated"` or user-supplied formulas are accepted.

type	A character string specifying one of the following approaches for performing adjusted model comparison tests: "boot" specifies a bootstrapping procedure; "rs2" specifies a Rao-Scott second-order adjustment; "all" specifies both approaches.
gof	A logical value indicating whether goodness-of-fit statistics should be calculated in addition to model comparison statistics. For model.HA = "saturated", model comparison statistics and goodness-of-fit statistics are identical, so only one set of statistics is presented.
print.status	A logical value indicating whether bootstrap progress updates should be provided.
...	Additional arguments passed to or from other methods.

Details

The Rao-Scott approach applies a second-order adjustment to the model comparison statistic and its sampling distribution. Formulas are provided in Appendix A of Bilder and Loughin (2007).

The bootstrap approach empirically estimates the sampling distribution of the model comparison statistic. Gange's (1995) method for generating correlated binary data is used for taking resamples under the null hypothesis. Bootstrap results are available only when boot = TRUE in the call to the genloglin function.

Value

— A list containing at least the following objects: original.arg and test.statistics.

original.arg is a list containing the following objects:

- model: The original model specified in the call to the genloglin function.
- model.HA: The alternative model specified for the model.HA argument.
- gof: The original value supplied to the gof argument.

test.statistics is a list containing at least the following objects:

- Pearson.chisq: The Pearson model comparison statistic calculated using the observed data.
- lrt: The LRT model comparison statistic calculated using the observed data.

If gof = TRUE, test.statistics additionally contains

- Pearson.chisq.gof: The Pearson goodness-of-fit statistic calculated using the observed data.
- lrt.gof: The LRT goodness-of-fit statistic calculated using the observed data.

— For type = "boot", the primary list additionally includes boot.results, a list containing at least the following objects:

- B.use: The number of bootstrap resamples used.
- B.discard: The number of bootstrap resamples discarded due to having at least one item with all positive or negative responses.
- p.chisq.boot: The bootstrap p-value for the Pearson model comparison test.

- p.lrt.boot: The bootstrap p-value for the LRT model comparison test.

If gof = TRUE, boot.results additionally contains

- p.chisq.gof.boot: The bootstrap p-value for the Pearson goodness-of-fit test.
- p.lrt.gof.boot: The bootstrap p-value for the LRT goodness-of-fit test.

— For type = "rs2", the primary list additionally includes rs.results, a list that includes at least Pearson.chisq.rs and lrt.rs.

Pearson.chisq.rs is a list containing the following objects:

- Pearson.chisq.rs: The Rao-Scott second-order adjusted Pearson model comparison statistic.
- df: The Rao-Scott second-order adjusted degrees of freedom for the model comparison statistic.
- p.value: The p-value for the Rao-Scott second-order adjusted Pearson model comparison test.

lrt.rs is a list containing the following objects:

- lrt.rs: The Rao-Scott second-order adjusted LRT model comparison statistic.
- df: Same as df given above.
- p.value: The p-value for the Rao-Scott second-order adjusted LRT model comparison test.

If gof = TRUE, rs.results additionally includes Pearson.chisq.gof.rs and lrt.gof.rs.

Pearson.chisq.gof.rs is a list containing the following objects:

- Pearson.chisq.gof.rs: The Rao-Scott second-order adjusted Pearson goodness-of-fit statistic.
- df: Same as df given above.
- p.value: The p-value for the Rao-Scott second-order adjusted Pearson goodness-of-fit test.

lrt.gof.rs is a list containing the following objects:

- lrt.gof.rs: The Rao-Scott second-order adjusted LRT goodness-of-fit statistic.
- df: Same as df given above.
- p.value: The p-value for the Rao-Scott second-order adjusted LRT goodness-of-fit test.

— For type = "all", the original list includes the boot.results and rs.results output.

References

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

Gange, S. (1995) Generating multivariate categorical variates using the iterative proportional fitting algorithm. *The American Statistician*, **49**, 134–138.

Examples

```
## For examples see help(genloglin).
```

`farmer1` *Data for One SRCV and One MRCV from the Kansas Farmer Survey*

Description

Responses for one SRCV and one MRCV from a survey of Kansas farmers. This data was first examined by Loughin and Scherer (1998) and subsequently examined by papers such as Agresti and Liu (1999) and Bilder, Loughin, and Nettleton (2000).

Usage

```
farmer1
```

Format

The data frame contains the following 6 columns:

Column 1, labeled `Ed`, corresponds to the respondent's highest attained level of education. A total of r = 5 levels of education are represented in the data.

- `1`: High school
- `2`: Vocational school
- `3`: 2-Year college
- `4`: 4-Year college
- `5`: Other

Columns 2-6 correspond to the response categories for the question "What are your primary sources of veterinary information?" Binary responses (1 = Yes, 0 = No) are provided for each category.

- `Y1`: Professional consultant
- `Y2`: Veterinarian
- `Y3`: State or local extension service
- `Y4`: Magazines
- `Y5`: Feed companies and representatives

Source

Loughin, T. M. and Scherer, P. N. (1998) Testing for association in contingency tables with multiple column responses. *Biometrics*, **54**, 630–637.

References

Agresti, A. and Liu, I.-M. (1999) Modeling a categorical variable allowing arbitrarily many category choices. *Biometrics*, **55**, 936–943.

Bilder, C., Loughin, T., and Nettleton, D. (2000) Multiple marginal independence testing for pick any/c variables. *Communications in Statistics–Theory and Methods*, **29**, 1285–1316.

Loughin, T. M. and Scherer, P. N. (1998) Testing for association in contingency tables with multiple column responses. *Biometrics*, **54**, 630–637.

`farmer2` *Data for Two MRCVs from the Kansas Farmer Survey*

Description

Responses for two MRCVs from a survey of Kansas farmers. This data was first examined by Bilder and Loughin (2007).

Usage

```
farmer2
```

Format

The data frame contains the following 7 columns:

Columns 1-3 correspond to the response categories for the question "Which of the following do you test your swine waste for?" Binary responses (1 = Yes, 0 = No) are provided for each category.

- `W1`: Nitrogen
- `W2`: Phosphorous
- `W3`: Salt

Columns 4-7 correspond to the response categories for the question "What swine waste disposal methods do you use?" Binary responses (1 = Yes, 0 = No) are provided for each category.

- `Y1`: Lagoon
- `Y2`: Pit
- `Y3`: Natural drainage
- `Y4`: Holding tank

Source

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

`farmer3` *Data for Three MRCVs from the Kansas Farmer Survey*

Description

Responses for three MRCVs from a survey of Kansas farmers. This data was first examined by Bilder and Loughin (2007).

Usage

```
farmer3
```

Format

The data frame contains the following 12 columns:

Columns 1-3 correspond to the response categories for the question "Which of the following do you test your swine waste for?" Binary responses (1 = Yes, 0 = No) are provided for each category.

- `W1`: Nitrogen
- `W2`: Phosphorous
- `W3`: Salt

Columns 4-7 correspond to the response categories for the question "What swine waste disposal methods do you use?" Binary responses (1 = Yes, 0 = No) are provided for each category.

- `Y1`: Lagoon
- `Y2`: Pit
- `Y3`: Natural drainage
- `Y4`: Holding tank

Columns 8-12 correspond to the response categories for the question "What are your primary sources of veterinary information?" Binary responses (1 = Yes, 0 = No) are provided for each category.

- `Z1`: Professional consultant
- `Z2`: Veterinarian
- `Z3`: State or local extension service
- `Z4`: Magazines
- `Z5`: Feed companies and representatives

Source

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

genloglin *Model the Association Among Two or Three MRCVs*

Description

The genloglin function uses a generalized loglinear modeling approach to estimate the association among two or three MRCVs. Standard errors are adjusted using a second-order Rao-Scott approach.

Usage

```
genloglin(data, I, J, K = NULL, model, add.constant = 0.5, boot = TRUE,
    B = 1999, B.max = B, print.status = TRUE)
```

Arguments

data	A data frame containing the raw data where rows correspond to the individual item response vectors, and columns correspond to the binary items, W1, ..., WI, Y1, ..., YJ, and Z1, ..., ZK (in this order).
I	The number of items corresponding to row variable W.
J	The number of items corresponding to column variable Y.
K	The number of items corresponding to strata variable Z.
model	For the two MRCV case, a character string specifying one of the following models: "spmi" (the complete independence model), "homogeneous" (the homogeneous association model), "w.main" (the w-main effects model), "y.main" (the y-main effects model), "wy.main" (the w-main and y-main effects model), or "saturated". Alternatively, a user-supplied formula can be specified, where the formula is limited to the generic variables W, Y, wi, yj, count, W1,..., WI, and Y1,..., YJ. For the three MRCV case, only user-supplied formulas are accepted. In addition to the generic variables defined for two MRCVs, the formula may include the generic variables Z, zk, and Z1,..., ZK.
add.constant	A positive constant to be added to all zero marginal cell counts.
boot	A logical value indicating whether bootstrap resamples should be taken.
B	The desired number of bootstrap resamples.
B.max	The maximum number of bootstrap resamples. Resamples for which at least one item has all positive or negative responses are thrown out; genloglin uses the first B valid resamples or all valid resamples if that number is less than B.
print.status	A logical value indicating whether progress updates should be provided. When print.status = TRUE, the status of the IPF algorithm is printed after every 5 iterations. Upon completion of the IPF algorithm, a progress bar appears that documents progress of the bootstrap.

Details

The genloglin function first converts the raw data into a form that can be used for estimation. For the two MRCV case, the reformatted data frame contains 2Ix2J rows and 5 columns generically named W, Y, wi, yj, and count. For the three MRCV case, the reformatted data frame contains 2Ix2Jx2K rows and 7 columns generically named W, Y, Z, wi, yj, zk, and count. Then, the model of interest is estimated by calling the glm function where the family argument is specified as poisson. For all predictor variables, the first level is the reference group (i.e., 1 is the reference for variables W, Y, and Z, and 0 is the reference for variables wi, yj, and zj).

The boot argument must equal TRUE in order to obtain bootstrap results with the genloglin method functions.

Value

— genloglin returns an object of class 'genloglin'. The object is a list containing at least the following objects: original.arg, mod.fit, sum.fit, and rs.results.

original.arg is a list containing the following objects:

- data: The original data frame supplied to the data argument.
- I: The original value supplied to the I argument.
- J: The original value supplied to the J argument.
- K: The original value supplied to the K argument.
- nvars: The number of MRCVs.
- model: The original value supplied to the model argument.
- add.constant: The original value supplied to the add.constant argument.
- boot: The original value supplied to the boot argument.

mod.fit is a list containing the same objects returned by glm with a few modifications as described in summary.genloglin.

sum.fit is a list containing the same objects returned by the summary method for class "glm" with a few modifications as described in summary.genloglin.

rs.results is a list containing the following objects (see Appendix A of Bilder and Loughin, 2007, for more detail):

- cov.mu: The covariance matrix for the estimated cell counts.
- E: The covariance matrix for the residuals.
- gamma: Eigenvalues used in computing second-order Rao-Scott adjusted statistics.

— For boot = TRUE, the primary list additionally includes boot.results, a list containing the following objects:

- B.use: The number of bootstrap resamples used.
- B.discard: The number of bootstrap resamples discarded due to having at least one item with all positive or negative responses.

- `model.data.star`: For the two MRCV case, a numeric matrix containing 2Ix2J rows and `B.use`+4 columns, where the first 4 columns correspond to the model variables `W`, `Y`, `wi`, and `yj`, and the last `B.use` columns correspond to the observed counts for each resample. For the three MRCV case, a numeric matrix containing 2Ix2Jx2K rows and `B.use`+6 columns, where the first 6 columns correspond to the model variables `W`, `Y`, `Z`, `wi`, `yj`, and `zk`, and the last `B.use` columns correspond to the observed counts for each resample.
- `mod.fit.star`: For the two MRCV case, a numeric matrix containing `B.use` rows and 2Ix2J +1 columns, where the first 2Ix2J columns correspond to the model-predicted counts for each resample, and the last column corresponds to the residual deviance for each resample. For the three MRCV case, a numeric matrix containing `B.use` rows and 2Ix2Jx2K+1 columns, where the first 2Ix2Jx2K columns correspond to the model-predicted counts for each resample, and the last column corresponds to the residual deviance for each resample.
- `chisq.star`: A numeric vector of length `B.use` containing the Pearson statistics (comparing `model` to the saturated model) calculated for each resample.
- `lrt.star`: A numeric vector of length `B.use` containing the LRT statistics calculated for each resample.
- `residual.star`: A numeric matrix with 2Ix2J rows (or 2Ix2Jx2K rows for the three MRCV case) and `B.use` columns containing the residuals calculated for each resample.

References

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

See Also

The `genloglin` methods `summary.genloglin`, `residuals.genloglin`, `anova.genloglin`, and `predict.genloglin`, and the corresponding generic functions `summary`, `residuals`, `anova`, and `predict`.

The `glm` function for fitting generalized linear models.

The `MI.test` function for testing for MMI (one MRCV case) or SPMI (two MRCV case).

Examples

```
# Estimate the y-main effects model for 2 MRCVs
mod.fit <- genloglin(data = farmer2, I = 3, J = 4, model = "y.main", boot = FALSE)
# Summarize model fit information
summary(mod.fit)
# Examine standardized Pearson residuals
residuals(mod.fit)
# Compare the y-main effects model to the saturated model
anova(mod.fit, model.HA = "saturated", type = "rs2")
# Obtain observed and model-predicted odds ratios
predict(mod.fit)

# Estimate a model that is not one of the named models
# Note that this was the final model chosen by Bilder and Loughin (2007)
```

```
mod.fit.other <- genloglin(data = farmer2, I = 3, J = 4, model = count ~ -1 + W:Y +
    wi%in%W:Y + yj%in%W:Y + wi:yj + wi:yj%in%Y + wi:yj%in%W3:Y1, boot =
    FALSE)
# Compare this model to the y-main effects model
anova(mod.fit, model.HA = count ~ -1 + W:Y + wi%in%W:Y + yj%in%W:Y + wi:yj +
    wi:yj%in%Y + wi:yj%in%W3:Y1, type = "rs2", gof = TRUE)

# To obtain bootstrap results from the method functions the genloglin() boot
# argument must be specified as TRUE (the default)
# A small B is used for demonstration purposes; normally, a larger B should be used
## Not run:
mod.fit <- genloglin(data = farmer2, I = 3, J = 4, model = "y.main", boot = TRUE,
    B = 99)
residuals(mod.fit)
anova(mod.fit, model.HA = "saturated", type = "all")
predict(mod.fit)

# Estimate a model for 3 MRCVs
mod.fit.three <- genloglin(data = farmer3, I = 3, J = 4, K = 5, model = count ~
    -1 + W:Y:Z + wi%in%W:Y:Z + yj%in%W:Y:Z + zk%in%W:Y:Z + wi:yj +
    wi:yj%in%Y + wi:yj%in%W + wi:yj%in%Y:W + yj:zk + yj:zk%in%Z +
    yj:zk%in%Y + yj:zk%in%Z:Y, boot = TRUE, B = 99)
residuals(mod.fit.three)
anova(mod.fit.three, model.HA = "saturated", type = "all")
predict(mod.fit.three, pair = "WY")
## End(Not run)
```

`item.response.table` *Create an Item Response Table or Data Frame*

Description

The `item.response.table` function is used to summarize data arising from one, two, or three MRCVs. For the one and two MRCV cases, a cross-tabulation of the positive and negative responses for each (Wi, Yj) pair is presented as a table or data frame (where Wi = W for the one MRCV case). For the three MRCV case, a cross-tabulation of the positive and negative responses for each (Wi, Yj) pair is presented conditional on the response for each Zk.

Usage

```
item.response.table(data, I, J, K = NULL, create.dataframe = FALSE)
```

Arguments

`data`	A data frame containing the raw data where rows correspond to the individual item response vectors, and columns correspond to the items W1, . . . , WI, Y1, . . . , YJ, and Z1, . . . , ZK (in this order).
`I`	The number of items corresponding to row variable W. I = 1 for the one MRCV case.

J The number of items corresponding to column variable Y.

K The number of items corresponding to strata variable Z.

create.dataframe
A logical value indicating whether the results should be presented as a data frame instead of a table.

Value

For create.dataframe = FALSE, item.response.table uses the tabular function (package **tables**) to produce tables of marginal counts.

For create.dataframe = TRUE, item.response.table returns the same information as above but presents it as a data frame. For the one MRCV case, the data frame contains rx2J rows and 4 columns generically named W, Y, yj, and count. For the two MRCV case, the data frame contains 2Ix2J rows and 5 columns named W, Y, wi, yj, and count. For the three MRCV case, the data frame contains 2Ix2Jx2K rows and 7 columns named W, Y, Z, wi, yj, zk, and count.

See Also

The marginal.table function for creating a marginal table that summarizes only the positive responses for each pair.

Examples

```
# Create an item response table for 1 SRCV and 1 MRCV
farmer.irtable.one <- item.response.table(data = farmer1, I = 1, J = 5)
farmer.irtable.one

# Create an item response data frame for 1 SRCV and 1 MRCV
farmer.irdataframe.one <- item.response.table(data = farmer1, I = 1, J = 5,
    create.dataframe = TRUE)
farmer.irdataframe.one

# Create an item response table for 2 MRCVs
farmer.irtable.two <- item.response.table(data = farmer2, I = 3, J = 4)
farmer.irtable.two

# Create an item response table for 3 MRCVs
farmer.irtable.three <- item.response.table(data = farmer3, I = 3, J = 4, K = 5)
farmer.irtable.three
```

---

marginal.table *Create a Marginal Table*

Description

The marginal.table function is used to summarize only the positive responses (joint positive responses for multiple MRCVs) for data arising from one, two, or three MRCVs. This function essentially provides a subset of counts from the item.response.table function.

Usage

```
marginal.table(data, I, J, K = NULL)
```

Arguments

data	A data frame containing the raw data where rows correspond to the individual item response vectors, and columns correspond to the items W1, ..., WI, Y1, ..., YJ, and Z1, ..., ZK (in this order).
I	The number of items corresponding to row variable W. I = 1 for the one MRCV case.
J	The number of items corresponding to column variable Y.
K	The number of items corresponding to strata variable Z.

Value

The `marginal.table` function uses the `tabular` function (package **tables**) to produce a table containing positive counts.

See Also

The `item.response.table` function for creating an item response table or data frame that summarizes both the positive and negative responses for each (Wi, Yj) pair (conditional on the response for each Zk in the case of three MRCVs).

Examples

```
# Create a marginal table for 1 MRCV
farmer.mtable.one <- marginal.table(data = farmer1, I = 1, J = 5)
farmer.mtable.one

# Create a marginal table for 2 MRCVs
farmer.mtable.two <- marginal.table(data = farmer2, I = 3, J = 4)
farmer.mtable.two

# Create a marginal table for 3 MRCVs
farmer.mtable.three <- marginal.table(data = farmer3, I = 3, J = 4, K = 5)
farmer.mtable.three
```

MI.test *Test for Marginal Independence*

Description

The `MI.test` function offers three approaches for testing multiple marginal independence (MMI) between one SRCV and one MRCV, or simultaneous pairwise marginal independence (SPMI) between two MRCVs.

Usage

```
MI.test(data, I, J, type = "all", B = 1999, B.max = B, summary.data =
    FALSE, add.constant = 0.5, plot.hist = FALSE, print.status = TRUE)

MI.stat(data, I, J, summary.data = FALSE, add.constant = 0.5)
```

Arguments

data	For summary.data = FALSE: a data frame containing the raw data where rows correspond to the individual item response vectors, and columns correspond to the items, W1, . . . , WI and Y1, . . . , YJ (in this order). For summary.data = TRUE: a data frame containing 4 columns generically named W, Y, yj, and count (one MRCV case), or 5 columns named W, Y, wi, yj, and count (two MRCV case).
I	The number of items corresponding to row variable W. I = 1 for the one MRCV case.
J	The number of items corresponding to column variable Y.
type	A character string specifying one of the following approaches for testing for MI: "boot" specifies a nonparametric bootstrap procedure; "rs2" specifies a Rao-Scott second-order adjustment; "bon" specifies a Bonferroni adjustment; "all" specifies all three approaches.
B	The desired number of bootstrap resamples.
B.max	The maximum number of bootstrap resamples. A resample is thrown out if at least one of the J (one MRCV case) or IxJ (two MRCV case) contingency tables does not have the correct dimension; MI.test uses the first B valid resamples or all valid resamples if that number is less than B.
summary.data	A logical value indicating whether data is a summary file containing the item response data instead of the raw data. Only type = "bon" is available for summary.data = TRUE.
add.constant	A positive constant to be added to all zero marginal cell counts.
plot.hist	A logical value indicating whether plots of the emprical bootstrap sampling distributions should be provided.
print.status	A logical value indicating whether bootstrap progress updates should be provided.

Details

The MI.test function calls MI.stat to calculate a modified Pearson statistic (see Bilder, Loughin, and Nettleton (2000) and Bilder and Loughin (2004)), and then performs the testing of MMI or SPMI. Three sets of testing methods are implemented:

- The nonparametric bootstrap resamples under the null hypothesis by independently sampling the W and Y vectors with replacement from the observed data. Fixed row counts (i.e., fixed counts for each level of the SRCV) are maintained for the one MRCV case. A modified Pearson statistic is calculated for each resample. In addition, bootstrap p-value combination methods are available to take advantage of the modified Pearson statistic's decomposition into

J (one MRCV case) or IxJ (two MRCV case) individual Pearson statistics. The minimum or the product of p-values is the combination for each resample.

- The Rao-Scott approach applies a second-order adjustment to the modified Pearson statistic and its sampling distribution. Formulas are provided in Appendix A of Bilder, Loughin, and Nettleton (2000) and Bilder and Loughin (2004). Note that this test can be conservative at times.
- The Bonferroni adjustment multiplies each p-value (using a standard chi-square approximation) from the individual Pearson statistics by J (one MRCV case) or IxJ (two MRCV case). If a resulting p-value is greater than 1 after the multiplication, it is set to a value of 1. The overall p-value for the test then is the minimum of these adjusted p-values. Note that the Bonferroni adjustment tends to produce an overly conservative test when the number of individual Pearson statistics is large.

Agresti and Liu (1999) discuss a marginal logit model approach that uses generalized estimation equations (GEE) to test for MMI. As shown in the example given below, this approach can be performed via functions available from the **geepack** package. However, Bilder, Loughin, and Nettleton (2000) caution that the Wald test produced by this approach does not hold the correct size, particularly when the sample size is not large and marginal probabilities are small.

Value

— `MI.test` returns a list containing at least `general`, a list containing the following objects:

- `data`: The original data frame supplied to the `data` argument.
- `I`: The original value supplied to the `I` argument.
- `J`: The original value supplied to the `J` argument.
- `summary.data`: The original value supplied to the `summary.data` argument.
- `X.sq.S`: The modified Pearson statistic; NA if at least one of the J (one MRCV case) or IxJ (two MRCV case) contingency tables does not have the correct dimension.
- `X.sq.S.ij`: A matrix containing the individual Pearson statistics.

— For `type = "boot"`, the primary list additionally includes `boot`, a list containing the following objects:

- `B.use`: The number of bootstrap resamples used.
- `B.discard`: The number of bootstrap resamples discarded due to having at least one contingency table with incorrect dimension.
- `p.value.boot`: The bootstrap p-value for the test of MMI or SPMI.
- `p.combo.min.boot`: The bootstrap p-value for the minimum p-value combination method.
- `p.combo.prod.boot`: The bootstrap p-value for the product p-value combination method.
- `X.sq.S.star`: A numeric vector containing the modified Pearson statistics calculated for each resample.
- `X.sq.S.ij.star`: A matrix containing the individual Pearson statistics calculated for each resample.
- `p.combo.min.star`: A numeric vector containing the minimum p-value calculated for each resample.

- `p.combo.prod.star`: A numeric vector containing the product p-value calculated for each resample.

— For `type = "rs2"`, the primary list additionally includes `rs2`, a list containing the following objects:

- `X.sq.S.rs2`: The Rao-Scott second-order adjusted Pearson statistic.
- `df.rs2`: The degrees of freedom for testing the second-order Rao-Scott adjusted Pearson statistic.
- `p.value.rs2`: The p-value based on the Rao-Scott second-order adjustment.

— For `type = "bon"`, the primary list additionally includes `bon`, a list containing the following objects:

- `p.value.bon`: The Bonferroni adjusted p-value for the test of MMI or SPMI.
- `X.sq.S.ij.p.bon`: A matrix containing the Bonferroni adjusted p-values for the individual Pearson statistics.

— For `type = "all"`, the list includes all of the above objects.

— `MI.stat` returns a list containing the following objects:

- `X.sq.S`: Defined above.
- `X.sq.S.ij`: Defined above.
- `valid.margins`: The number of contingency tables with correct dimension.

References

Agresti, A. and Liu, I.-M. (1999) Modeling a categorical variable allowing arbitrarily many category choices. *Biometrics*, **55**, 936–943.

Bilder, C. and Loughin, T. (2004) Testing for marginal independence between two categorical variables with multiple responses. *Biometrics*, **36**, 433–451.

Bilder, C., Loughin, T., and Nettleton, D. (2000) Multiple marginal independence testing for pick any/c variables. *Communications in Statistics–Theory and Methods*, **29**, 1285–1316.

See Also

The `genloglin` function offers a generalized loglinear modeling approach for testing the relationship among two or three MRCVs.

Examples

```
# Test for MMI using the second-order Rao-Scott adjustment
test.mmi.rs2 <- MI.test(data = farmer1, I = 1, J = 5, type = "rs2")
test.mmi.rs2

# Test for MMI using all three approaches
# A small B is used for demonstration purposes; normally, a larger B should be used
## Not run:
test.mmi.all <- MI.test(data = farmer1, I = 1, J = 5, type = "all", B = 99,
```

```
    plot.hist = TRUE)
test.mmi.all
## End(Not run)

# Use MI.test() with summary data
# Convert raw data file to summary file for this example
farmer1.irdframe <- item.response.table(data = farmer1, I = 1, J = 5, create.dataframe =
    TRUE)
# Test for MMI using the Bonferroni adjustment
test.mmi.bon <- MI.test(data = farmer1.irdframe, I = 1, J = 5, type = "bon",
    summary.data = TRUE)
test.mmi.bon

# Test for SPMI using the second-order Rao-Scott adjustment
test.spmi.rs2 <- MI.test(data = farmer2, I = 3, J = 4, type = "rs2")
test.spmi.rs2

# Test for MMI using the marginal logit model approach
## Not run:
library(geepack)
n<-nrow(farmer1)
farmer1.id<-cbind(case=1:n, farmer1)
# Reshape raw data into long format as required by geeglm() function
# Assumes 3:ncol(farmer1.id) corresponds to MRCV items
farmer1.gee<-reshape(data = farmer1.id,
                varying = names(farmer1.id)[3:ncol(farmer1.id)],
                v.names = "response", timevar = "item", idvar = "case",
                direction = "long")
row.names(farmer1.gee)<-NULL
farmer1.gee[,2:3]<-lapply(farmer1.gee[,2:3], factor)
# Data frame must be ordered by case
farmer1.gee<-farmer1.gee[order(farmer1.gee$case),]
head(farmer1.gee)
tail(farmer1.gee)
mod.fit.H0<-geeglm(formula = response ~ item, family = binomial(link = logit),
                  data = farmer1.gee, na.action = na.omit, id = case,
                  corstr = "unstructured")
mod.fit.HA<-geeglm(formula = response ~ Ed*item, family = binomial(link = logit),
                   data = farmer1.gee, na.action = na.omit, id = case,
                   corstr = "unstructured")
# Compute Wald test
anova(mod.fit.HA, mod.fit.H0)
## End(Not run)
```

`predict.genloglin` *Calculate Observed and Model-Predicted Odds Ratios for MRCV Data*

Description

The `predict.genloglin` method function calculates observed and model-predicted odds ratios and their confidence intervals using results from `genloglin`. It offers an asymptotic normal approxima-

tion for estimating the confidence intervals for the observed and model-predicted odds ratios, and a bootstrap approach for estimating the confidence intervals for the model-predicted odds ratios.

Usage

```
## S3 method for class 'genloglin'
predict(object, alpha = 0.05, pair = "WY", print.status = TRUE, ...)
```

Arguments

object	An object of class 'genloglin' produced by the genloglin function.
alpha	The desired alpha level. The predict.genloglin function provides two-sided (1-alpha)x100% confidence intervals.
pair	For the case of three MRCVs, a character string specifying the pair of items for which odds ratios will be calculated: "WY" indicates odds ratios should be calculated for each (Wi, Yj) pair conditional on the response for each Zk, "WZ" indicates conditional odds ratios should be calculated for each (Wi, Zk) pair, and "YZ" indicates conditional odds ratios should be calculated for each (Yj, Zk) pair.
print.status	A logical value indicating whether bootstrap progress updates should be provided.
...	Additional arguments passed to or from other methods.

Details

Wald confidence intervals are estimated for both model-based (see Appendix A of Bilder and Loughin, 2007) and observed (see Agresti, 2013, p. 70) odds ratios.

A bootstrap method is also available which provides bias-corrected accelerated (BCa) confidence intervals for the model-predicted odds ratios. See Efron (1987) for more information about BCa intervals. The predict.genloglin function uses a jackknife approximation for estimating the empirical influence values.

The bootstrap confidence intervals are available only when boot = TRUE in the original call to the genloglin function.

Value

— A list containing at least original.arg, OR.obs, and OR.model.asymp.

original.arg is a list containing the following objects:

- data: The original data frame supplied to the data argument.
- I: The original value supplied to the I argument.
- J: The original value supplied to the J argument.
- K: The original value supplied to the K argument.
- nvars: The number of MRCVs.
- alpha: The original value supplied to the alpha argument.

`OR.obs` is a numeric matrix. For the two MRCV case, the matrix contains IxJ rows corresponding to the IxJ possible pairs (Wi, Yj) and 3 columns, where column 1 corresponds to the observed odds ratio for (Wi, Yj) and columns 2 and 3 correspond to the estimated lower and upper confidence bounds, respectively. For the three MRCV case, the matrix contains 2xIxJxK rows corresponding to all possible combinations of `pair` conditional on the response for each item of the 3rd MRCV, and 3 columns as described for the 2 MRCV case.

`OR.model.asymp` is a numeric matrix similar to `OR.obs` but where column 1 corresponds to the model-predicted odds ratios and columns 2 and 3 correspond to the estimated lower and upper confidence bounds, respectively, using an asymptotic normal approximation.

— For `boot = TRUE` in the call to the `genloglin` function, the primary list additionally includes `boot.results`, a list containing the following objects:

- `B.use`: The number of bootstrap resamples used.
- `B.discard`: The number of bootstrap resamples discarded due to having at least one item with all positive or negative responses.
- `OR.model.BCa`: A numeric matrix similar to `OR.obs` but where column 1 corresponds to the model-predicted odds ratios and columns 2 and 3 correspond to the estimated lower and upper confidence bounds, respectively, of the BCa intervals.

References

Agresti, A. (2013) *Categorical data analysis (3rd ed.)*. Hoboken, New Jersey: John Wiley & Sons.

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

Efron, B. (1987) Better bootstrap confidence intervals. *Journal of the American Statistical Association*, **82**, 171–185.

Examples

```
## For examples see help(genloglin).
```

`print.genloglin` *Control Printed Display of MRCV Regression Modeling Objects*

Description

Method functions that control the printed display of MRCV regression modeling objects.

Usage

```
## S3 method for class 'genloglin'
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'summary.genloglin'
print(x, digits = max(3, getOption("digits") - 3), symbolic.cor =
    x$symbolic.cor, signif.stars = getOption("show.signif.stars"), ...)
```

```
## S3 method for class 'anova.genloglin'
print(x, ...)
## S3 method for class 'predict.genloglin'
print(x, ...)
```

Arguments

x	An object of class "genloglin" produced by the genloglin function.
digits	Minimum number of digits; see print.default for additional explanation.
symbolic.cor	A logical value indicating whether correlations should be printed in symbolic form; see the summary method for class "glm" for additional explanation.
signif.stars	A logical value indicating whether significance stars should be printed; see the summary method for class "glm" for additional explanation.
...	Additional arguments passed to or from other methods.

Details

The print.genloglin function is based on the print method for class "glm".

The print.summary.genloglin function is based on the print method for class "summary.glm".

Value

A print out of selected results.

---

print.MMI *Control Printed Display of Objects of Class "MMI"*

Description

The print.MMI method function controls the printed display of objects of class MMI.

Usage

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

Arguments

x	An object of class "MMI" produced by the MI.test function.
...	Additional arguments passed to or from other methods.

Value

A print out of selected results from MI.test.

print.SPMI *Control Printed Display of Objects of Class "SPMI"*

Description

The `print.SPMI` method function controls the printed display of objects of class `SPMI`.

Usage

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

Arguments

`x`	An object of class `"SPMI"` produced by the `MI.test` function.
`...`	Additional arguments passed to or from other methods.

Value

A print out of selected results from `MI.test`.

residuals.genloglin *Calculate Standardized Pearson Residuals for MRCV Data*

Description

The `residuals.genloglin` method function calculates standardized Pearson residuals for the model specified in the `genloglin` function. It offers an asymptotic approximation and a bootstrap approximation for estimating the variance of the residuals.

Usage

```
## S3 method for class 'genloglin'
residuals(object, ...)
```

Arguments

`object`	An object of class `'genloglin'` produced by the `genloglin` function.
`...`	Additional arguments passed to or from other methods.

Details

The bootstrap results are only available when `boot = TRUE` in the call to the `genloglin` function.

The `residuals.genloglin` function uses `tabular` (package **tables**) to display the results for the two MRCV case.

See Bilder and Loughin (2007) for additional details about calculating the residuals.

Value

— A list containing at least `std.pearson.res.asymp.var`. For the two MRCV case, the object is a 2Ix2J table of class `'tabular'` containing the standardized Pearson residuals based on the estimated asymptotic variance. For the three MRCV case, the object is a data frame containing the 2Ix2Jx2K residuals.

— For `boot = TRUE` in the call to the `genloglin` function, the list additionally includes:

- `B.use`: The number of bootstrap resamples used.
- `B.discard`: The number of bootstrap resamples discarded due to having at least one item with all positive or negative responses.
- `std.pearson.res.boot.var`: For the two MRCV case, a 2Ix2J table of class `'tabular'` containing the standardized Pearson residuals based on the bootstrap variance. For the three MRCV case, a data frame containing the 2Ix2Jx2K residuals.

References

Bilder, C. and Loughin, T. (2007) Modeling association between two or more categorical variables that allow for multiple category choices. *Communications in Statistics–Theory and Methods*, **36**, 433–451.

Examples

```
## For examples see help(genloglin).
```

`sealion` *Steller Sea Lion Scat Data of Riemer, Wright, and Brown (2011)*

Description

The sealion data frame is a portion of the data analyzed by Riemer, Wright, and Brown (2011). The data represents the types of prey found in Steller sea lion scats found at the mouth of the Columbia river in August 2004 and 2007. The goal was to determine if the diet habits of the sea lions had changed over time.

Usage

```
sealion
```

Format

The data frame contains the following 12 columns:

Column 1, labeled `Date`, corresponds to the date of data collection. Two collection dates, 8/10/2004 and 8/7/2007, are included in the data frame.

Columns 2-12 correspond to the types of prey found in Steller sea lion scats. Binary responses (1 = Present in scat, 0 = Not present in scat) are provided for each category.

- prey1: Unidentified fish
- prey2: Pacific lamprey (*Lampetra tridentata*)
- prey3: Starry flounder (*Platichthys stellatus*)
- prey4: Pacific sardine (*Sardinops sagax*)
- prey5: Pacific herring (*Clupea pallasii*)
- prey6: Unidentified clupeid (family Clupeidae)
- prey7: Unidentified skate (family Rajidae)
- prey8: Northern anchovy (*Engraulis mordax*)
- prey9: Pacific salmon (*Oncorhynchus* spp.)
- prey10: Pacific staghorn sculpin (*Leptocottus armatus*)
- prey11: Pacific hake (*Merluccius productus*)

Source

Riemer, S. D., Wright, B. E., and Brown, R. F. (2011) Food habits of Steller sea lions (*Eumetopias jubatus*) off Oregon and northern California, 1986-2007. *Fishery Bulletin*, **109**, 369–381.

---

summary.genloglin *Summarize Two or Three MRCV Model Fit Information*

Description

The summary.genloglin function summarizes model fit information provided by the genloglin function.

Usage

```
## S3 method for class 'genloglin'
summary(object, ...)
```

Arguments

object	An object of class 'genloglin' produced by the genloglin function.
...	Additional arguments passed to or from other methods.

Details

The summary.genloglin function is based on the summary method for class "glm" with a few modifications. The coefficients object contains Rao-Scott second-order adjusted standard errors, z-values, and p-values. The cov.unscaled object contains the Rao-Scott second-order adjusted covariance matrix of the estimated coefficients.

The deviance information printed by summary.genloglin should not be used to conduct traditional model comparison tests. The anova.genloglin function offers adjusted tests.

Value

The `summary.genloglin` function returns the same list returned by the `summary` method for class `"glm"` with the exception of AIC.

Examples

```
## For examples see help(genloglin).
```

Index