Package 'cvam'

October 1, 2021

Type Package

Title Coarsened Variable Modeling

Version 0.9.1

Date 2021-10-01

Author Joseph L. Schafer <Joseph.L.Schafer@census.gov>

Maintainer Joseph L. Schafer <Joseph.L.Schafer@census.gov>

Description Extends R's implementation of categorical variables (factors) to handle coarsened observations; implements log-linear models for coarsened categorical data, including latent-class models. Detailed information and examples are provided in the package vignettes.

Depends R (>= 3.3.0)

Imports stats, Formula, coda

Suggests nnet, lme4, MASS, xtable

License GPL-3 | file LICENSE

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-10-01 13:50:02 UTC

R topics documented:

`cvam-package` *Coarsened Variable Modeling*

Description

Extends R's implementation of categorical variables (factors) to handle coarsened observations; implements log-linear models for coarsened categorical data, including latent-class models. Detailed information and examples are provided in the package vignettes.

Details

Log-linear models, when applied to frequencies in a multiway table, describe associations among the factors that define the dimensions of the table. Standard functions for fitting log-linear models in R, including `glm` and `loglin`, cannot accept observations with incomplete information on any of the factors in the model, because those observations cannot be assigned with certainty to a single cell of the complete-data table. The functions in the `cvam` package facilitate log-linear modeling of factors with missing and coarsened values. The two major functions are:

```
coarsened     Create a coarsened factor
cvam          Log-linear models for coarsened factors
```

A coarsened factor is an extended version of a factor whose elements may be fully observed, partially observed or missing. The partially-observed and missing states are represented by extra levels which are interpreted as groupings of the fully observed states. The `cvam` function fits log-linear models to coarsened factors. It also accepts ordinary factors with or without missing values, and factors that contain only missing values, which are useful for latent-class analysis. The modeling routines implemented in `cvam` include EM algorithms for mode finding and Markov chain Monte Carlo procedures for Bayesian simulation and multiple imputation. Supporting funtions are used to extract information from a fitted model, including:

```
anova          Compare the fit of cvam models
cvamEstimate   Estimated marginal and conditional probabilities
cvamPredict    Predict missing or coarsened values
cvamLik        Likelihood of observed data patterns
cvamImpute     Impute missing or coarsened values
```

Five datasets are also provided:

```
abortion2000        Abortion attitudes from the General Social Survey
crime               Crime victimization data
hivtest             HIV test dataset
microUCBAdmissions  U.C. Berkeley graduate admissions microdata
seatbelt            Seatbelt data
```

Author(s)

Joseph L. Schafer <Joseph.L.Schafer@census.gov>

Maintainer: Joseph L. Schafer <Joseph.L.Schafer@census.gov>

References

Extended descriptions and examples for all major functions are provided in two vignettes, *Understanding Coarsened Factors in cvam* and *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

`abortion2000` *Abortion Attitudes from the 2000 General Social Survey*

Description

This dataset, which was extracted from the 2000 General Social Survey (GSS) (Smith et al., 2019), reports the responses of adults in the United States to seven questions about legalized abortion. The questions began, "Please tell me whether or not you think it should be possible for a pregnant woman to obtain a legal abortion if..." The abortion items were given to a random two-thirds subsample of GSS participants, so about 33% of the values are `NA` by design. Refusal to answer the question (a rare occurrence) was also coded here as `NA`.

The data frame also includes variables on age, sex, race, Hispanic origin, education, religious and political affiliation. A second race item, which was modeled after the race question on the U.S. Census questionnaire, was given to a random half-sample.

Usage

```
abortion2000
```

Format

a data frame with 2,817 rows and 16 factors:

`Age` respondent's age, with levels `"18-29"`, `"30-49"`, `"50-64"`, and `"65+"`

`Sex` respondent's sex, coded as `"Female"` or `"Male"`

`Race` respondent's race, coded as `"White"`, `"Black"`, or `"Other"`; see NOTE below

`CenRace` respondent's race, coded as `"White"`, `"Black"`, `"Hisp"` or `"Other"`; see NOTE below

`Hisp` respondent's Hispanic classification, with levels `"nonHisp"` and `"Hisp"`

`Degree` respondent's education, classified as `"<HS"` (did not finish high school), `"HS"` (high school diploma), `"JunCol"` (junior college), `"Bach"` (Bacheor's degree), or `"Grad"` (graduate degree)

`Relig` respondent's religious preference, classified as `"Prot"` (Protestant), `"Cath"` (Roman Catholic), `"Jewish"`, `"None"`, or `"Other"`

`Party` respondent's political party identification, with levels `"Dem"` (Democrat), `"Rep"` (Republican), and `"Ind/Oth"` (Independent or Other); see NOTE below

`PolViews` respondent's political views, with levels `"Con"` (Conservative), `"Mod"` (Moderate), and `"Lib"` (Liberal)

Each of the seven variables below is a factor with levels `"Yes"`, `"No"`, and `"DK"` (don't know). The items were prefixed by, "Please tell me whether or not you think it should be possible for a pregnant woman to obtain a legal abortion if..."

`AbDefect` "...If there is a strong chance of serious defect in the baby?"

`AbNoMore` "...If she is married and does not want any more children?"

`AbHealth` "...If the woman's own health is seriously endangered by the pregnancy?"

`AbPoor` "...If the family has a very low income and cannot afford any more children?"

`AbRape` "...If she became pregnant as a result of rape?"

`AbSingle` "...If she is not married and does not want to marry the man"

`AbAny` "...The woman wants it for any reason?"

Note

`Race`, which corresponds to the GSS variable `race`, is based on the interviewer's assessment of the respondent's race. When interviewers were not sure, they could ask the respondent, "What race do you consider yourself?"

`CenRace` is a collapsed version of the GSS variable `racecen1`. That variable, which was modeled on the race question in the U.S. Census, was given to half of the GSS sample in 2000 and to the full sample in subsequent years. Participants could choose from over a dozen race categories, or they could select "Some other race" and provide their own. The `"Hisp"` values represent those who chose "Some other race" and described themselves as Hispanic, Latino, Latina, or something similar.

`Party` is based on the GSS variable `partyid`. Level `"Dem"` includes Decocrat-leaning Independents, and `"Rep"` includes Republican-leaning Independents.

Source

Smith, T.W., Davern, M., Freese, J., and Morgan, S.L. (2019) *General Social Surveys, 1972–2018*. National Data Program for the Social Sciences, No. 25., 1 data file (64,814 logical records) + 1 codebook (3,758 pp.). Chicago: NORC.

anova.cvam *Comparing the Fit of Two or More Models*

Description

Compares the fit of two or more cvam objects

Usage

```
## S3 method for class 'cvam'
anova(object, ..., method = c("lrt", "logP", "AIC", "BIC"),
   pval = FALSE, pvalDigits = 4L, showRank=NULL )
```

Arguments

object	an object produced by cvam
...	additional cvam objects
method	criterion for model comparison: "lrt" uses -2 times the loglikelihood function for a standard likelihood-ratio test; "logP" uses -2 times the penalized loglikelihood or log-posterior density; "AIC" uses Akaike information criterion; and "BIC" uses Bayesian information criterion.
pval	if TRUE then p-values will be computed if method is "lrt" or "logP".
pvalDigits	digits for rounding of p-values
showRank	if TRUE, models will be ranked from best to worst (with rank 1 being the best) according to the fit measure specified by method. Defaults to TRUE if method is "AIC" or "BIC"

Details

The p-values reported for the "lrt" and "logP" methods use a standard chi-squared approximation, with degrees of freedom equal to the difference in the number of parameters for the models being compared. This approximation is valid only if the models being compared are properly nested and ordered, with the simplest model appearing first in the argument list. The chi-squared approximation can be poor if the degrees of freedom for the comparison is large, and if the model corresponding to the null hypothesis (i.e., the smaller one) has fitted cell means that are too small. The chi-squared approximation is not appropriate for comparing latent-class models with a different number of latent classes.

The likelihood function used in "lrt" and "logP" is based on a Poisson model for the cell means in the complete-data table. The Poisson model is an appropriate surrogate for a multinomial model. It is also an appropriate surrogate for a product multinomial if the model includes all possible associations among variables regarded as fixed.

The residual degrees of freedom are the difference between the number of free parameters in a saturated Poisson model minus the number of free parameters in the current Poisson model. The

saturated model estimates one parameter for every cell in the complete-data table, excluding dimensions for latent factors, and excluding structural-zero cells. No adjustments are made for estimates on a boundary of the parameter space.

For `"BIC"`, the sample size is taken to be the total number of observations or total frequency in the data supplied by the user to fit the model, which does not include a flattening constant or any nuggets from a prior distribution created by `cvamPrior`. No adjustments are made for missing or coarsened values.

Value

an object of class `c("anova","data.frame")`

Author(s)

Joe Schafer `<Joseph.L.Schafer@census.gov>`

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

`cvam`

Examples

```
M0 <- cvam( ~ V1 + V2, data=crime, freq=n )
M1 <- cvam( ~ V1 * V2, data=crime, freq=n )
anova(M0, M1, pval=TRUE)
```

---

baseLevels *Get Coarsened Factor Attributes*

Description

Retrieve specific attributes of a coarsened factor.

Usage

```
baseLevels(x)
nBaseLevels(x)
coarseLevels(x)
nCoarseLevels(x)
mapping(x)
```

Arguments

`x`	a coarsened factor

Details

A coarsened factor, produced by the function coarsened, is an extended type of factor whose elements may be fully observed, partially observed, or missing. The full set of attributes of a coarsened factor may be obtained by attributes, and individual attributes are available with attr. The functions documented on this page are convenient alternatives to attr.

The elements of baseLevels, a character vector of length nBaseLevels, represent states of complete knowledge. The elements of coarseLevels, a character vector of length nCoarseLevels, represent states of incomplete or missing information. Each element of coarseLevels maps onto two or more elements of baseLevels. The attribute mapping is an integer matrix with nCoarseLevels rows and nBaseLevels columns, with 1 in position [i,j] if coarse level i contains base level j. The last coarse level is always NA, and it contains every base level.

Value

The requested attribute of x.

Note

A coarsened factor has the usual attributes of a factor, but they should not be altered directly. For example, the function levels<-, the replacement version of levels, should not be used with a coarsened factor.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Understanding Coarsened Factors in cvam*.

See Also

cvam, coarsened, is.naCoarsened, dropCoarseLevels

Examples

```
fac <- factor( c("red", "green", NA, "yellow", "notRed", "green") )
cFac <- coarsened( fac,
   levelsList = list("notRed" = c("green", "yellow")) )
baseLevels(cFac)
mapping(cFac)
```

coarsened *Coarsened Factors*

Description

A coarsened factor is an extended version of a factor or ordered factor whose elements may be fully observed, partially observed or missing. The partially-observed and missing states are represented by extra levels which are interpreted as groupings of the fully observed states. Coarsened factors are specifically designed for modeling with the cvam package.

Usage

```
coarsened(obj, levelsList = list(), warnIfCoarsened = TRUE)

is.coarsened(x)

## S3 method for class 'coarsened'
print(x, quote = FALSE, max.levels = NULL,
   width = getOption("width"), ...)

## S3 method for class 'coarsened'
droplevels(x, ...)

## S3 method for class 'coarsened'
relevel(x, ...)

## S3 method for class 'coarsened'
reorder(x, ...)

## S3 method for class 'coarsened'
rep(x, ...)

## S3 method for class 'coarsened'
x[...]

## S3 method for class 'coarsened'
x[[...]]

## S3 replacement method for class 'coarsened'
x[...] <- value

## S3 replacement method for class 'coarsened'
x[[...]] <- value
```

Arguments

obj	a factor or ordered factor to be converted to a coarsened factor

levelsList	a named list that defines the groupings of levels(obj) to indicate states of partial knowledge
warnIfCoarsened	if TRUE, a warning is issued if obj is already a coarsened factor
x	a coarsened factor or other object
quote	logical, indicating whether or not strings should be printed with surrounding quotes
max.levels	integer, indicating how many base levels and coarse levels should be printed for a coarsened factor; if 0, no extra base levels or coarse levels lines will be printed. The default, NULL, entails choosing max.levels such that the base levels and coarse levels each print on one line of width width
width	only used when max.levels is NULL; see above
...	additional arguments passed to or from other methods
value	character: a set of levels for replacement

Details

A coarsened factor, which inherits from class "factor" or c("ordered","factor"), has two types of levels: base levels, which represent states of complete knowledge, and coarse levels, which represent states of incomplete knowledge. Each coarse level maps to two or more base levels. The mapping is defined by the argument levelsList.

For example, consider a factor whose levels are c("red","notRed","green","yellow"), where "notRed" denotes an observation that is either "green" or "yellow". When the factor is converted to a coarsened factor, c("red","green","yellow") becomes the baseLevels, and "notRed" becomes an element of coarseLevels. To produce this result, the argument levelsList should have a component named "notRed", whose value is c("green","yellow").

The last coarse level is NA, denoting an observation that could belong to any of the base levels. The NA coarse level is created automatically. Calling coarsened with an empty levelsList (the default) produces a coarsened factor with NA as its only coarse level.

If the main argument to coarsened is already a coarsened factor, then a warning is issued (if warnIfCoarsened is TRUE) and the coarsened factor is returned unchanged.

The generic functions droplevels, relevel, and reorder should not be applied to coarsened factors; the S3 methods droplevels.coarsened, relevel.coarsened, and reorder.coarsened will prevent this from happening.

rep.coarsened is a method for the generic function rep that ensures the special attributes of a coarsened factor are preserved.

Extraction and replacement methods `[` and `[[` are also provided to preserve the special attributes of coarsened factors.

Value

coarsened returns a coarsened factor.

is.coarsened returns TRUE if x is a coarsened factor and FALSE otherwise.

Note

Coarsened factors were designed for use by the modeling function cvam, which treats base levels and coarse levels differently. Other statistical modeling routines, such as lm, may not handle them appropriately. Functions outside of the cvam package will treat coarse levels (including NA) the same as base levels, producing results that are difficult to interpret or nonsensical, especially if the base levels are ordered.

The behavior of coarsened with levelsList = list() is similar to that of addNA, which converts the missing values in a factor to non-missing observations with value NA and adds NA to the levels. The result of addNA, however, is an ordinary factor or ordered factor which has no mechanism to inform other functions that NA has special meaning.

The function is.na should not be applied to a coarsened factor; use is.naCoarsened instead.

Because base levels and coarse levels should be handled differently, functions from base R that manipulate the levels of a factor, including relevel, reorder, droplevels, and the replacement version of levels should not be used with coarsened factors. Supplying a coarsened factor to any of these functions will produce an error.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Understanding Coarsened Factors in cvam*.

See Also

cvam, is.naCoarsened, baseLevels, dropCoarseLevels

Examples

```
fac <- factor( c("red", "green", NA, "yellow", "notRed", "green") )
cFac <- coarsened( fac,
   levelsList = list("notRed" = c("green", "yellow")) )
print(cFac)
# extraction and replacement
print( cFac[2:3] )
cFac[2:3] <- c("NA", "green")
```

---

crime *Crime Victimization Data*

Description

This dataset, reported and analyzed by Kadane (1985), comes from the National Crime Survey conducted by the U.S. Bureau of the Census. Occupants of housing units were interviewed to determine whether they had been victimized by crime in the preceding six-month period. Six months later, the units were visited again to determine whether the occupants had been victimized during the intervening months. Missing values for various reasons occurred at both occasions.

Usage

```
crime
```

Format

a data frame with 9 rows and 3 variables:

`V1` victimization status from the first visit

`V2` victimization status from the second visit

`n` frequency in sample

Source

Kadane, J.B. (1985) Is victimization chronic? a Bayesian analysis of multinomial missing data. *Journal of Econometrics*, 29, 47-67.

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

---

cvam *Log-Linear Models for Incomplete Categorical Variables*

Description

Fits log-linear models to categorical variables by three methods: maximizing the loglikelihood or log-posterior density by Expectation-Maximization (EM) algorithms, simulating the posterior distribution by a Markov chain Monte Carlo (MCMC) algorithms, and creating random draws of parameters from an approximate Bayesian posterior distribution. The factors in the model may have missing or coarsened values.

Usage

```
cvam(obj, ...)

## S3 method for class 'formula'
cvam(obj, data, freq, prior = cvamPrior(),
    method = c("EM", "MCMC", "approxBayes", "mfSeen", "mfTrue",
       "mfPrior", "modelMatrix"), control = list(), omitData = FALSE,
    saturated = FALSE, modelMatrix = NULL, offset = NULL,
    strZero = NULL, startVal = NULL, estimate = NULL, ...)

## S3 method for class 'cvam'
cvam(obj, method = obj$method, control = NULL, startVal = NULL,
    estimate = NULL, ...)
```

Arguments

obj	an object used to select a method: either a model formula or the result from a previous call to cvam.
data	an optional data frame, list or environment (or object coercible to a data frame by as.data.frame) containing the variables in the model. If not found in data, the variables are taken from environment(obj), typically the environment from which cvam is called.
freq	an optional variable for holding integer frequencies when the observations are grouped. If freq is not given, then the observations are assumed to represent microdata, and all frequencies are set to one.
prior	an object produced by cvamPrior to represent prior information incorporated into the model fit.
method	a procedure for fitting the model: "EM" computes a maximum-likelihood (ML) estimate, penalized ML estimate, or posterior mode; "MCMC" runs a Markov chain Monte Carlo algorithm to simulate a sequence of correlated random draws from the posterior distribution of the unknown parameters; "approxBayes" creates independent draws from an approximate posterior distribution. The other alternatives return various objects without fitting the model.
control	a named list containing control parameters which are passed to cvamControl. Control parameters determine the maximum number of iterations, criteria for judging convergence, proposal distributions for MCMC, and so on. Control parameters that are not found in this list are set to default values.
omitData	if TRUE, then the observations supplied through data and freq are ignored, and the fitted model is based only the prior information supplied through prior. Combining omitData=TRUE with method="MCMC" will simulate random draws from the prior distribution.
saturated	if TRUE, then a saturated model is fit to the cell means without defining a model matrix or log-linear coefficients.
modelMatrix	an optional model matrix that defines the log-linear model. In ordinary circumstances, cvam creates the model matrix automatically by interpreting terms in the model formula and referring to the contrast attributes of the model factors. In rare circumstances, a user may want to supply a different model matrix. The model matrix should have one row for every cell in the complete-data table. If a model matrix is supplied, the model formula is used only to identify the variables that are included the model, not to define the associations among them.
offset	an optional numeric vector of length NROW(modelMatrix) containing an offset for the log-linear model. If omitted, the offset is assumed to be zero for every cell.
strZero	an optional logical vector of length NROW(modelMatrix) containing TRUE for every cell to be considered a structural zero and FALSE elsewhere. Structural zeros are assumed to have zero probability and are omitted from the model fitting. If strZero is omitted, all elements are assumed to be FALSE.
startVal	an optional vector of starting values for the model parameters. If saturated=FALSE, this should be a vector of length NCOL(modelMatrix) containing log-linear coefficients; if saturate=FALSE, it should be a vector of length NROW(modelMatrix)

containing cell probabilities or cell means, which are automatically rescaled to become probabilities.

`estimate` an optional formula or list of formulas of the kind expected by `cvamEstimate` specifying marginal or conditional probabilities to be estimated, bypassing the need for a subsequent call to that function.

`...` values to be passed to the methods.

Details

A log-linear model is specified by a one-sided formula that determines which associations among the variables are allowed. For example, `~ A + B + C` implies that `A`, `B` and `C` are mutually independent; `~ A*B + A*C` implies that `B` and `C` are conditionally independent given `A`; and so on. Variables in a model may be factors or coarsened factors, and missing values are permitted. All models are fit using a surrogate Poisson formulation which is appropriate for Poisson, multinomial or product-multinomial sampling. A formula may contain a vertical bar to specify variables to be regarded as fixed; for example, `~ A*B + A*C | A` fixes the variable `A`. Fixing variables does not change the model fitting procedure; the only difference is that, after the model has been fit, the cell probabilities are scaled to sum to one within every combination of levels of the fixed variables.

If `cvam` is called with a `cvam` object as its first argument, then the data, model and prior distribution will be taken from the previous run, and (unless `startVal` is supplied), starting values will be set to the final parameter values from the previous run.

Value

if `method` is `"EM"`, `"MCMC"` or `"approxBayes"`, an object of class `c("cvam","list")` containing the results of a model fit. For other values of `method`, the requested object is returned without fitting a model.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

Extended descriptions and examples for all major functions are provided in two vignettes, *Understanding Coarsened Factors in cvam* and *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

`coarsened`, `cvamPrior`, `cvamControl`, `cvamEstimate`, `get.coef`, `summary.cvam`

Examples

```
# convert U.C. Berkeley admissions three-way table to data frame,
# fit model of conditional independence, display summary
# compare the fit to the saturated model
dF <- as.data.frame(UCBAdmissions)
fit <- cvam( ~ Dept*Gender + Dept*Admit, data=dF, freq=Freq )
```

```
summary(fit)
fitSat <- cvam( ~ Dept*Gender*Admit, data=dF, freq=Freq )
anova(fit, fitSat, pval=TRUE)

# fit non-independence model to crime data; then run MCMC for
# 5000 iterations, creating 10 multiple imputations of the frequencies
# for the 2x2 complete-data table
fit <- cvam( ~ V1 * V2, data=crime, freq=n )
set.seed(56182)
fitMCMC <- cvam(fit, method="MCMC",
   control=list( iterMCMC=5000, imputeEvery=500) )
get.imputedFreq(fitMCMC)
```

`cvamControl` *Control Parameters for* `cvam`

Description

The `cvam` function fits log-linear models to coarsened categorical variables. Its model-fitting procedures are governed by parameters in a `cvamControl` object created by the auxiliary function documented here. This function is intended for internal use; the only reason to invoke this function directly is to display the control parameters and their default values.

Usage

```
cvamControl( iterMaxEM = 500L, iterMaxNR = 50L,
   iterApproxBayes = 1L, imputeApproxBayes = FALSE,
   iterMCMC = 5000L, burnMCMC = 0L, thinMCMC = 1L, imputeEvery = 0L,
   saveProbSeries = FALSE,
   typeMCMC = c("DA","RWM"), tuneDA = c(10,.8,.8), tuneRWM = c(1000,.1),
   stuckLimit = 25L,
   startValDefault = c("center", "uniform"), startValJitter = 0,
   critEM = 1e-06, critNR = 1e-06, critBoundary = 1e-08, ncolMaxMM = 1000L,
   excludeAllNA = TRUE, critModelCheck=1e-08, confidence=.95,
   probRound = TRUE, probDigits = 4L )
```

Arguments

`iterMaxEM`	maximum number of iterations performed when `method = "EM"`; see DETAILS.
`iterMaxNR`	maximum number of iterations of Newton-Raphson performed during an M-step of EM; see DETAILS.
`iterApproxBayes`	number of simulated log-linear coefficient vectors to be drawn from their approximate posterior distribution when `method="approxBayes"`.
`imputeApproxBayes`	if `TRUE` then, for each draw of the log-linear coefficients from their approximate posterior distribution, the true frequencies will be imputed.

iterMCMC	number of iterations of Markov chain Monte Carlo after the burn-in period when method="MCMC".
burnMCMC	number of iterations of Markov chain Monte Carlo performed as a burn-in period, for which the reults are discarded. The total number of iterations performed is burnMCMC+iterMCMC.
thinMCMC	thinning interval for saving the results from MCMC as a series.
imputeEvery	imputation interval for saving imputed frequencies for the complete-data table. If 0, then no imputations are saved.
saveProbSeries	if TRUE then the simulated values of cell probabilities from MCMC will be stored as a series.
typeMCMC	either "DA" (data augmentation) or "RWM" (random-walk Metropolis); see DETAILS.
tuneDA	tuning parameters for data augmentation MCMC; see DETAILS.
tuneRWM	tuning parameter for random-walk Metropolis MCMC; see DETAILS.
stuckLimit	criterion for deciding if the MCMC algorithm has gotten stuck.
startValDefault	method used to obtain default starting values for parameters if no starting values are provided. "center" begins in the center of the parameter space, which assigns equal probability to all non-structural zero cells in the complete-data table. "uniform" draws random starting values from a uniform distribution on the cell probabilities.
startValJitter	standard deviation for Gaussian random noise added to starting values. If cvam is called with saturated=FALSE, the log-linear coefficients are perturbed by this amount; if saturated=TRUE, the log-cell probabilities are perturbed by this amount and renormalized to sum to one.
critEM	convergence criterion for EM stopping rule; see DETAILS.
critNR	convergence criterion for Newton-Raphson stopping rule in M-step of EM; see DETAILS.
critBoundary	criterion for testing whether estimated probabilities are close to zero, in which case a warning is given.
ncolMaxMM	limit on the number of columns allowed for a log-linear model matrix.
excludeAllNA	if TRUE, then cases for which all modeled variables are missing will be excluded from the model fitting procedure, because they only contribute constant terms to the observed-data loglikelihood function.
critModelCheck	criterion for checking the log-linear model matrix for linear dependencies among the columns.
confidence	confidence coefficient for interval estimates, used when estimates are requested in the call to cvam.
probRound	if TRUE, estimated probabilities will be rounded.
probDigits	number of digits for rounding estimated probabilities.

Details

When cvam is called with method="EM", it performs an EM algorithm. At each E-step, observations with missing or coarsened values are apportioned to cells of the complete-data table in the expected amounts determined by the current estimated parameters. At the M-step, the a log-linear model is fit to the predicted complete-data frequencies from the E-step, using a Newton-Raphson procedure if saturated=FALSE. The EM algorithm is stopped after iterMaxEM iterations, or when the maximum absolute difference in cell means from one iteration to the next is no greater than critEM. The Newton-Raphson procedure in each M-step is stopped after iterMaxNR iterations or when the maximum absolute difference in cell means from one iteration to the next is no greater than critNR.

When cvam is called with method="MCMC", the algorithm that is run depends on typeMCMC and on whether the model is fit with saturated=TRUE.

- If saturated=FALSE and typeMCMC="DA", then the algorithm is a data-augmentation procedure that resembles EM. At each cycle, observations with missing or coarsened values are randomly allocated to cells of the complete-data table by drawing from a multinomial distribution, and the log-linear coefficients are updated using one step of a Metropolis-Hastings algorithm that mimics Newton-Raphson and conditions on the allocated frequencies. The proposal distribution is multivariate-t and can be adjusted by tuning constants in tuneDA, a numeric vector containing the degrees of freedom, step size and scale factor.
- If saturated=FALSE and typeMCMC="RWM", the observations with missing or coarsened values are not allocated, and the log-linear coefficients are updated by a step of random-walk Metropolis. The proposal is mutivariate-t and can be adjusted by tuning constants in tuneRWM, a numeric vector containing the degrees of freedom and scale factor.
- If saturated=TRUE, then the algorithm is a data-augmentation procedure that requires no tuning.

Full details on the EM and MCMC procedures are given in the Appendix of the vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

Value

a list of control parameters for internal use by the function cvam.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

See Also

cvam

Examples

```
# display all control parameters and their default values
cvamControl()
```

cvamEstimate *Obtain Estimated Probabilities from a Fitted Model*

Description

After fitting a log-linear model with cvam, the fitted model object may be passed to this function to obtain estimated marginal and conditional probabilities for model factors.

Usage

```
cvamEstimate(estimate, obj, meanSeries = TRUE,
   confidence = obj$control$confidence,
   probRound = obj$control$probRound, probDigits = obj$control$probDigits, ...)

## S3 method for class 'cvamEstimate'
print(x, showHeader = TRUE, ...)

## S3 method for class 'cvamEstimateList'
print(x, showHeader = TRUE, ...)
```

Arguments

estimate	a formula or list of formulas indicating the desired probabilities; see DETAILS.
obj	an object produced by cvam containing results from a model fit
meanSeries	applies when obj contains results from a simulation run. If TRUE, then the requested estimates are computed based on a running mean of cell probabilities over all iterations after the burn-in period. If FALSE, then the requested estimates are based only on the cell probabilities from the final iteration, and (assuming the run was sufficiently long, if it is MCMC) can be regarded as a single draw from their posterior distribution.
confidence	confidence coefficient for asymmetric interval estimates; see DETAILS.
probRound	if TRUE, probabilities will be rounded.
probDigits	number of digits for rounding probabilities.
x	a set of estimates to be printed.
showHeader	if TRUE, a descriptive header is printed.
...	additional arguments to be passed to print.

Details

The argument estimate should be a one-sided formula or a list of one-sided formulas, with variables separated by '+', and variables to be conditioned on appearing after '|'. For example, ~ A requests marginal probabilities for every level of A; ~ A + B | C + D requests conditional probabilities for every level combination of A and B given every level combination of C and D.

- If obj was produced with saturated=FALSE and method="EM", then standard errors for all probabilities are computed using Taylor linearization, also known as the delta method, based on the asymptotic covariance matrix for the log-linear coefficients.
- If obj was produced with saturated=FALSE and method="MCMC" or "approxBayes", then standard errors are computed with Taylor linearization, based on the covariance matrix for the simulated log-linear coefficients from all iterations after the burn-in period.
- If obj was produced with saturated=TRUE, then standard errors are not computed.

A symmetric confidence interval for a probability may be problematic, especially if the estimate is close to zero or one. Asymmetric confidence intervals are computed by applying a normal approximation on the logistic (log-odds) scale and translating the endpoints back to the probability scale.

Value

if estimate is a single formula, this function returns a data frame containing estimated probabilities, standard errors, and endpoints of approximate confidence intervals. If estimate is a list of formulas, then a list of data frames is returned.

Note

Estimates may also be requested when fitting a model with cvam, by providing a formula or list of formulas to the optional argument estimate.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

cvam cvamPredict cvamImpute cvamLik

Examples

```
fit <- cvam( ~ Sex * PolViews * AbAny, data=abortion2000 )
cvamEstimate( list( ~ AbAny | Sex, ~ AbAny | PolViews ), fit )
```

cvamImpute *Impute Data from a Fitted Model*

Description

After fitting a log-linear model with cvam, the fitted model object may be passed to this function, along with a dataset containing missing or coarsened values, to randomly impute the true data from their predictive distribution given the observed data and parameters from the fitted model.

Usage

```
cvamImpute(obj, data, freq, meanSeries = FALSE, synthetic=FALSE)
```

Arguments

obj	an object produced by cvam containing results from a model fit
data	data frame for imputation, possibly different from the data used to fit the model contained in obj
freq	variable containing frequencies for data. If omitted, all frequencies are taken to be 1, meaning that the imputation frame is assumed to contain microdata.
meanSeries	applies when obj contains results from a simulation run. If TRUE, then the imputations are based on a running mean of cell probabilities over all iterations after the burn-in period. If FALSE, then the imputations are based only on the cell probabilities from the final iteration, and (assuming the run was sufficiently long, if MCMC) can be regarded as a single draw from their posterior distribution. See DETAILS below.
synthetic	if TRUE, then observed values for all variables in the data frame (excluding variables that are conditioned on in the model and regarded as fixed) are set to NA and imputed, producing a dataset that is fully synthetic.

Value

a data frame containing imputed data. If freq was given, the data frame has one row for each cell in the complete-data table and a variable freq containing the frequencies. If freq was not given, the data frame has one row for each microdata observatiom.

Note

When this function is used within a process for multiple imputation, meanSeries should be set to FALSE, otherwise the imputations will not correctly reflect uncertainty about model parameters.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

cvam, cvamEstimate, cvamPredict, cvamLik

Examples

```
# impute from a grouped dataset with frequencies
fit <- cvam( ~ V1 * V2, freq=n, data=crime )
cvamImpute( fit, data=crime, freq=n )
# impute microdata
fit <- cvam( ~ Sex * PolViews * AbAny, data=abortion2000 )
impData <- cvamImpute( fit, data=abortion2000 )
head(impData)
```

---

cvamLik *Likelihood of Observed Data Patterns*

Description

After fitting a log-linear model with cvam, the fitted model object may be passed to this function, along with a dataset that may contain missing or coarsened values, to compute the likelihood of each pattern of possibly incomplete or coarsened data for subset of variables, possibly conditioned upon another subset of variables

Usage

```
cvamLik(form, obj, data, meanSeries = TRUE)
```

Arguments

form	a formula indicating which variables to consider, and which variables to condition on, when computing the likelihood
obj	an object produced by cvam containing results from a model fit
data	data frame for computing the likelihood values, possibly different from the data used to fit the model contained in obj
meanSeries	applies when obj contains results from a simulation run. If TRUE, then the requested likelihood values are based on a running mean of cell probabilities over all iterations after the burn-in period. If FALSE, then the requested values are based only on the cell probabilities from the final iteration, and (assuming the run was sufficiently long, if MCMC) can be regarded as a single draw from their posterior distribution.

Details

For structural zeros, `0/0` is returned as `0`. If any variables are being conditioned on in `form`, they must not contain any missing or coarsened values.

Value

A data frame containing the model variables, with a variable `likVall` holding the likelihood values

Author(s)

Joe Schafer `<Joseph.L.Schafer@census.gov>`

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

`cvam`, `cvamEstimate`, `cvamImpute`, `cvamPredict`

Examples

```
result <- cvam( ~ V1 * V2, freq=n, data=crime)
cvamLik( ~ V1 + V2, result, data=crime )
```

`cvamPredict` *Predict Missing or Coarsened Values from a Fitted Model*

Description

After fitting a log-linear model with `cvam`, the fitted model object may be passed to this function, along with a dataset containing missing or coarsened values, to predict one or more variables from their predictive distribution given the observed data and parameters from the fitted model.

Usage

```
cvamPredict(form, obj, data, freq, meanSeries = TRUE, sep = ".")
```

Arguments

`form`	a one-sided formula indicating the variable or variables to be predicted, with variables separated by '+'
`obj`	an object produced by `cvam` containing results from a model fit
`data`	data frame for prediction, possibly different from the data used to fit the model contained in `obj`

freq	variable containing frequencies for data. If omitted, all frequencies are taken to be 1, meaning that the prediction frame is assumed to contain microdata.
meanSeries	applies when obj contains results from a simulation run. If TRUE, then the requested predictions are based on a running mean of cell probabilities over all iterations after the burn-in period. If FALSE, then the requested predictions are based only on the cell probabilities from the final iteration, and (assuming the run was sufficiently long, if MCMC) can be regarded as a single draw from their posterior distribution.
sep	character sting used to separate the levels of multiple variables being predicted

Details

Predictions from this function are unlike predictions from a regression model. In regression, prediction is to compute the estimated mean response ar specific values of the predictors. With this function, predictions are based on the predictive distribution for one or more variables given all the observed data, including the variable(s) to be predicted if they are seen. The prediction for a variable that is seen will assign a probability of one to the seen value and zero probability to other values.

Value

A data frame containing the predicted probabilities or frequencies, with an attribute colFrame that identifies its columns

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

cvam, cvamEstimate, cvamImpute, cvamLik

Examples

```
fit <- cvam( ~ V1 + V2, freq=n, data=crime )
cvamPredict( ~ V1, fit, data=crime, freq=n )   # predict frequencies
cvamPredict( ~ V1, fit, data=crime  )          # predict probabilities
```

cvamPrior *Data-Augmentation Prior for Coarsened Factor Loglinear Model*

Description

The cvam function fits loglinear models to coarsened categorical variables. The cvamPrior function creates an object to pass to cvam to represent prior information that is incorporated into the model fit.

Usage

```
cvamPrior(obj = list(), flatten = 0, ridge = 0, intensity = 1)
```

Arguments

obj	a list of prior information nuggets to apply to the complete-data frequency table; see DETAILS.
flatten	a prior information nugget to be divided equally across all cells of the complete-data frequency table; see DETAILS.
ridge	a ridge factor to apply to the log-linear coefficients; see DETAILS.
intensity	a factor applied simultaneously to all prior information to scale it up or down; see DETAILS.

Details

An object produced by this function, when passed to cvam through its prior argument, incorporates prior information as

- a flattening constant, a positive value that is divided equally among all non-structural zero cells of the complete-data table, and
- prior nuggets, which take the form of coarsened-data frequencies that are assigned to selected cells or groups of cells.

Log-linear models fit with saturated=FALSE can also accept a ridge factor, which acts upon the coefficients in a manner similar to ridge regression, shrinking the estimated coefficients toward zero and stabilizing its estimated covariance matrix. The added information is equivalent to a multivariate normal prior density centered at zero with prior precision (inverse covariance) matrix equal to the ridge factor times the identity matrix.

The intensity factor provides a simple way to strengthen or weaken the overall amount of prior information, which is useful for sensitivity analyses. The flattening constant, nugget frequencies and ridge factor are all multiplied by intensity. Setting intensity=2 doubles the prior information, intensity=.5 cuts it in half, and so on.

Value

an object of class "cvamPrior", designed for use by the function cvam.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

cvam, coarsened

Examples

```
# fit a saturated model to a four-way table
fit <- cvam( ~ Sex*CenRace*Hisp*Party, data=abortion2000,
 saturated=TRUE )
# add a flattening constant
fit <- cvam( ~ Sex*CenRace*Hisp*Party, data=abortion2000,
 saturated=TRUE, prior=cvamPrior( flatten=10 ) )

# fit with saturated=FALSE and no prior information, and
# notice how large the SEs are
fit <- cvam( ~ Sex*CenRace*Hisp*Party, data=abortion2000,
 saturated=FALSE )
head( get.coef(fit, withSE=TRUE) )
# add a very mild ridge factor and notice how the SEs
# have become reasonable
fit <- cvam( ~ Sex*CenRace*Hisp*Party, data=abortion2000,
 saturated=FALSE, prior=cvamPrior( ridge=.1 ) )
head( get.coef(fit, withSE=TRUE) )

# add s few prior nuggets to stabilize the distribution
# of Party within a rare category
nuggetList <- list(
   list( CenRace="Black", Hisp="Hisp", Party="Dem", freq=1 ),
   list( CenRace="Black", Hisp="Hisp", Party="Rep", freq=1 ),
   list( CenRace="Black", Hisp="Hisp", Party="Ind/Oth", freq=1 ) )
myPrior <- cvamPrior( nuggetList, flatten=10 )
summary(myPrior)
fit <- cvam( ~ Sex*CenRace*Hisp*Party, data=abortion2000,
 saturated=FALSE, prior=myPrior )
```

---

dropCoarseLevels *Remove Coarse Levels from a Coarsened Factor*

Description

A coarsened factor, produced by the function coarsened, has two types of levels: base levels, to represent observations that are fully known, and coarse levels, to represent observations that are partially or completely missing. The function dropCoarseLevels converts a coarsened factor to a factor or ordered factor by removing all of the coarse levels and setting the corresponding observations to NA.

Usage

```
dropCoarseLevels(x)
```

Arguments

x	a factor or coarsened factor

Details

If the only coarse level of x is NA, then no information is lost when dropCoarseLevels is applied. If x has other non-empty coarse levels, then the partial information carried by those observations is effectively discarded.

Value

A factor or ordered factor, obtained by removing the coarse levels of x and setting the observations in those levels to NA.

If x is a factor but not a coarsened factor, then it is returned unchanged.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information about coarsened factors in the cvam package, see the vignette *Understanding Coarsened Factors in cvam*.

See Also

cvam, coarsened, baseLevels, is.naCoarsened

Examples

```
fac <- factor( c("red", "green", NA, "yellow", "notRed", "green") )
cFac <- coarsened( fac,
   levelsList = list("notRed" = c("green", "yellow")) )
dropCoarseLevels(cFac)
```

get.coef *Extract Information from a Coarsened-Variable Model*

Description

This group of functions will extract various summaries from a model fit by cvam, either with the EM algorithm or by Markov chain Monte Carlo.

Usage

```
get.coef(obj, withSE = FALSE, meanSeries = TRUE, msgIfNone = TRUE)

get.covMat(obj, msgIfNone = TRUE)

get.estimates(obj, msgIfNone = TRUE)

get.loglik(obj, msgIfNone = TRUE)

get.logP(obj, msgIfNone = TRUE)

get.mfTrue(obj)

get.modelMatrix(obj, msgIfNone = TRUE)

get.offset(obj, mfTrue = FALSE, msgIfNone = TRUE)

get.strZero(obj, mfTrue = FALSE, msgIfNone = TRUE)

get.fitted(obj, type=c("prob", "mean", "logMean"), mfTrue = TRUE,
   meanSeries = TRUE, msgIfNone = TRUE )

get.imputedFreq(obj, msgIfNone = TRUE)

get.minus2logPSeries(obj, startValShift = TRUE,
   msgIfNone = TRUE, coda = ( obj$method == "MCMC" ) )

get.coefSeries(obj, msgIfNone = TRUE, coda = ( obj$method == "MCMC" ) )

get.probSeries(obj, levelNames=TRUE, sep=".",
msgIfNone = TRUE, coda = ( obj$method == "MCMC" ) )
```

Arguments

obj	an object resulting from a call to cvam with method = "EM" or method = "MCMC"

withSE — if TRUE, then get.coef will return a data frame containing estimated log-linear coefficients, standard errors, t-statistics and p-values; if FALSE, then only a vector of coefficients is given.

mfTrue — if TRUE, then get.offset, get.strZero and get.fitted will return a data frame with one row per cell, with all model variables (the non-coarsened versions) present as factors, and with another variable named (depending on which function was called) offset, strZero or fit containing the requested values. If FALSE, then get.offset, get.strZero and get.fitted will return a vector containing the requested values.

meanSeries — applies when obj is the result from a simulation run. If TRUE, results will be based on from a running average of simulated parameters over all iterations after the burn-in period. If FALSE, results will be based only on the simulated parameter values at the end of the run.

msgIfNone — if TRUE then, if the get procedure fails, an informative message is given explaining why the requested summary cannot be obtained. For example, get.coef will fail to return coefficients from a model fit with cvam(...,saturated = TRUE) because no model matrix is created and the log-linear coefficients are not defined. If FALSE, then these messages are suppressed.

type — type of fitted values to be returned by get.fitted. "prob" returns cell probabilities conditioned on variables fixed by the model (if any); "mean" returns cell means from the log-linear model; and "logMean" returns log-cell means from the log-linear model (i.e., the linear predictor).

startValShift — the function get.minus2logPSeries extracts a saved series from an MCMC run containing the values of (minus 2 times) the log-posterior density function. If startValShift is true, the series is shifted by (minus 2 times) the log-posterior density at the starting value, if the starting value appears to be a mode.

coda — if TRUE, the series from an MCMC run is returned as an mcmc object for plotting and diagnostic analysis with the coda package. If FALSE, a one-dimensional series is returned as a numeric vector, and a multidimensional series is returned as a numeric matrix with rows corresponding to iterations and columns corresponding to elements of the multidimensional quantities being monitored.

levelNames — the get.probSeries function extracts a saved series of probabilities from an MCMC run corresponding to cells of the complete-data table (i.e., the rows of mfTrue). If levelNames is TRUE, names for the cell probabilities are constructed from the levels of the factors in mfTrue. As the number of variables in the model grows, these names can become unwieldly, and setting levelNames to FALSE omits the names.

sep — a character string used to separate the levels of multiple factors when levelNames is TRUE.

Details

The series objects returned by get.minus2logPSeries, get.coefSeries and get.probSeries omit results from the burn-in period, if any, and may also be thinned. The default behavior is no burn-in period and no thinning. The burn-in period and thinning interval are set by components

of the control argument to cvam, via the function cvamControl; the relevant components are control$burnMCMC and control$thinMCMC. By default, cvam does not save cell probabilities. To save them, set control$saveProbSeries to TRUE.

get.imputedFreq returns multiple imputations of frequencies for the complete-data table generated and stored during an MCMC run after the burn-in period. The default behavior is no imputation. This can be changed by setting control$imputeEvery to an integer greater than zero.

Other useful information from a model fit can be extracted with the summary method for a cvam object, and with the functions cvamEstimate, cvamPredict, cvamLik, and cvamImpute.

Value

get.coef returns a vector of estimated coefficients from the log-linear model if withSE=FALSE; if withSE=TRUE, it returns a data frame containing coefficients, standard errors, t-statistics and p-values.

get.covMat returns an estimated covariance matrix for the estimated coefficients.

get.estimates returns a data frame or a list of data frames containing the estimates held in obj.

get.loglik and get.logP return the value of the loglikelihood function or log-posterior density from the beginning of the final iteration of EM or MCMC. If the model was fit using cvam(...,saturated=TRUE), the likelihood is based on a multinomial or product-multinomial distribution over the cells of the complete-data table. If the model was fit as a log-linear approach using cvam(...,saturated=FALSE), the likelihood is based on a surrogate Poisson model.

get.mfTrue returns a data frame with one row per cell of the complete-data table. The variables in this data frame include every factor appearing in the model (the non-coarsened versions) and another variable named freq. If the model was fit using cvam(...,method="EM"), freq contains the predicted cell frequencies at the final iteration of EM. If the model was fit using cvam(...,method="MCMC"), freq contains a running average of imputed cell frequencies over all iterations of MCMC after the burn-in period. In either case, if the data used to fit the model contain no missing or coarsened values, then freq will be equal to the observed frequencies.

get.modelMatrix returns the model matrix for the log-linear model. The rows of the model matrix correspond to the rows of mfTrue, and the columns correspond to terms created from the factors in mfTrue.

get.offset retrieves the offset for the log-linear model. If mfTrue is TRUE, it returns the data frame mfTrue with a numeric variable named offset. If mfTrue is FALSE, it returns a numeric vector of length NROW(mfTrue).

get.strZero retrieves the logical values indicating whether each cell is structural zero. If mfTrue is TRUE, it returns the data frame mfTrue with a logical variable named strZero. If mfTrue is FALSE, it returns a logical vector of length NROW(mfTrue).

get.fitted retrieves fitted values from the log-linear model. If type="prob", the fitted values are cell probabilities conditioned on any variables fixed in the model. If type="mean" or "logMean", the fitted values are cell means or log-cell means from the log-linear model. If mfTrue is TRUE, the function returns the data frame mfTrue with a numeric variable named fit. If mfTrue is FALSE, it returns a numeric vector of length NROW(mfTrue).

get.imputedFreq returns the data frame mfTrue, with the freq variable replaced by multiply imputed versions of the frequencies for the complete-data table.

`get.minus2logPSeries` returns a series of (minus 2 times) the log-posterior density values from the iterations of MCMC, either as a numeric vector or as an `mcmc` object used by the `coda` package.

`get.coefSeries` returns a series of log-linear coefficients from the iterations of MCMC, either as a numeric matrix or as an `mcmc` object used by the `coda` package.

`get.probSeries` returns a series of cell probabilities from the iterations of MCMC, either as a numeric matrix or as an `mcmc` object used by the `coda` package.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

For information about `coda`, see:

Martyn Plummer, Nicky Best, Kate Cowles and Karen Vines (2006). CODA: Convergence Diagnosis and Output Analysis for MCMC, *R News*, vol 6, 7-11.

See Also

`cvam`, `summary.cvam`, `cvamControl`, `cvamEstimate`, `cvamPredict`, `cvamImpute`, `cvamLik`

Examples

```
fit <- cvam( ~ V1 * V2, data=crime, freq=n )
get.coef(fit, withSE=TRUE)
get.covMat(fit)
get.fitted(fit, type="mean")

set.seed(6755)
fit <- cvam(fit, method="MCMC",
   control=list(iterMCMC=5000, imputeEvery=500) )
get.imputedFreq(fit)

## Not run: plot( get.coefSeries(fit) )  # coda trace and density plots
```

---

hivtest *HIV test dataset*

Description

This dataset concerns the diagnostic accuracy of tests for HIV infection. Four different tests (`A`, `B`, `C`, and `D`) were applied to 428 high-risk patients. The result from each test is either negative (`"neg"`) or positive (`"pos"`). Yang and Becker (1997) applied latent-class analysis to these data to estimate the sensitivity and specificity of these tests in the absence of a gold standard.

Usage

```
hivtest
```

Format

a data frame with 9 rows and 5 variables:

`A` test result from radioimmunoassay (RIA) utilizing recombinant agl21

`B` test result from RIA utilizing purified HIV p24

`C` test result from RIA utilizing purified HIV gpl20

`D` test result from enzyme-linked immunosorbent assay

`COUNT` frequency; number of patients exhibiting the given pattern of results

Source

Yang, I and Becker, M.P. (1997) Latent variable modeling of diagnostic accuracy. *Biometrics*, 53, 948-958.

---

is.naCoarsened *Missing-Value Indicators for Coarsened Factors*

Description

A coarsened factor, produced by the function `coarsened`, stores missing values differently from an ordinary factor. If the base R function `is.na` is applied to a coarsened factor, every element of the result will be `FALSE`. The function `is.naCoarsened` is the suitable alternative to `is.na` for coarsened factors.

Usage

```
is.naCoarsened(x)
```

Arguments

`x`	a coarsened factor

Details

A coarsened factor, produced by the function `coarsened`, has two types of levels. Base levels represent states of complete information, and coarse levels represent states of incomplete or missing information. Each coarse level maps onto two or more base levels. The last coarse level is `NA`, which maps onto every base level.

Value

A logical vector of the same length as `x`, with `TRUE` indicating that an element of `x` is `NA`, and `FALSE` otherwise.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information about coarsened factors in the cvam package, see the vignette *Understanding Coarsened Factors in cvam*.

See Also

cvam, coarsened, baseLevels, dropCoarseLevels

Examples

```
fac <- factor( c("red", "green", NA, "yellow", "notRed", "green") )
cFac <- coarsened( fac,
   levelsList = list("notRed" = c("green", "yellow")) )
is.naCoarsened(cFac)
```

---

latentFactor *Latent Factor*

Description

A latent factor is a categorical variable whose values are entirely missing. The function latentFactor provides a convenient way to create a latent factor with a given number of base levels, which is useful for latent-class modeling with cvam.

Usage

```
latentFactor( n, levels = 2L )

is.latentFactor(x)
```

Arguments

n	length of the factor
levels	either an integer specifying the number of base levels, or a character vector containing labels for the base levels
x	an object to be tested

Value

For latentFactor, a latent coarsened factor of length n; for is.latentFactor, TRUE or FALSE.

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignettes *Understanding Coarsened Factors in cvam* and *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

cvam, coarsened, baseLevels, is.naCoarsened

Examples

```
# fit latent class model to hivtest data
hivtest$L <- latentFactor( NROW(hivtest), 2 )
set.seed(125)
fit <- cvam( ~ L*A + L*B + L*C + L*D, data=hivtest, freq=COUNT,
   control=list(startValJitter=.1) )
cvamEstimate( list( ~L, ~A|L, ~B|L, ~C|L, ~D|L ), fit )
```

---

microUCBAdmissions *UC Berkeley Graduate Admissions Microdata*

Description

The dataset UCBAdmissions, distributed with the R datasets package, is a three-dimensional array that classifies applicants to graduate school at the University of California, Berkeley in 1973 by sex, department, and admission status. That table, first published by Bickel *et al.* (1975), is frequently used in textbooks and statistics courses to illustrate Simpson's paradox. This dataset, microUCBAdmissions is a simulated data frame with three factors and one row per individual; when the factors are tabulated, they reproduce the frequencies in UCBAdmissions.

Usage

```
microUCBAdmissions
```

Format

a data frame with 4,526 rows and 3 factors:

Admit "Admitted" or "Rejected"

Gender "Male" or "Female"

Dept "A", "B", "C", "D", "E", or "F"

Source

Bickel, P.J., Hammel, E.A. and O'Connell, J.W. (1975) Sex bias in graduate admissions: Data from Berkeley. *Science*, 187, 398-403.

`miInference` *Combine results from analyses after multiple imputation*

Description

This function combines the results from data analyses performed after multiple imputation using methods described by Rubin (1987) and others.

Usage

```
miInference( est.list, std.err.list, method = "scalar",
   df.complete = NULL )

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

Arguments

`est.list`	a list of estimates to be combined. Each component of this list should be a scalar or vector containing point estimates from the analysis of an imputed dataset. This list should have M components, where M is the number of imputations, and all components should have the same length.
`std.err.list`	a list of standard errors to be combined. Each component of this list should be a scalar or vector of standard errors associated with the estimates in `est.list`.
`method`	how are the estimates to be combined? At present, the only type allowed is `"scalar"`, which means that estimands are treated as one-dimensional entities. If `est.list` contains vectors, inference for each element of the vector is carried out separately; covariances among them are not considered.
`df.complete`	degrees of freedom assumed for the complete-data inference. This should be a scalar or a vector of the same length as the components of `est.list` and `std.err.list`.
`x`	a result from `miInference`.
`...`	values to be passed to the methods.

Details

If `df.complete = NULL` or `Inf`, the degrees of freedom are computed by the method of Rubin (1987, Chap.3), which assumes that if there were no missing data, the usual normal approximation for large samples would be appropriate, i.e. that a 95% interval would be computed as the estimate plus or minus 1.96 standard errors. Otherwise, the degrees of freedom are computed by the method of Barnard and Rubin (1999), which assumes that an approximate 95% interval without missing data would be the estimate plus or minus `qt(.975,df.complete)` standard errors.

The result from this function is a list whose `class` attribute has been set to `"miInference"`. If this list is displayed or printed via the generic function `print`, it will be formatted into a table

resembling the output from a regression analysis with columns for the estimates, standard errors, t-ratios (estimates divided by their standard errors) and p-values for testing the null hypothesis that each estimate is zero.

Value

a list with the following components:

`names`	character-string labels for the estimands. This is derived from the `names` attribute, if any, of the components of `est.list`.
`est`	combined estimate(s).
`std.err`	standard error(s) for `est`.
`df`	degrees of freedom for Student-t approximation. For example, 95% intervals can be computed as `est` plus or minus `qt(.975,df)*std.err`.
`p`	p-value(s) for testing the null hypothesis that each estimand is zero against a two-tailed alternative.
`rel.incr`	estimated relative increase(s) in variance due to nonresponse.
`mis.inf`	estimated rate(s) of missing information.

Note

Rubin (1987) defined the rate of missing information as `rel.incr + 2/(df+3)` divided by `(rel.incr+1)`, which estimates the information lost due to missing values and due to the fact that the number of multiple imputations is finite. We define it as `rel.incr` divided by `(rel.incr+1)`, the information lost due to missing values, which is consistent with the formulas of Barnard and Rubin (1999).

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

Barnard, J. and Rubin, D.B. (1999) Small-sample degrees of freedom with multiple imputation. *Biometrika*, 86, 948-955.

Rubin, D.B. (1987) *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Examples

```
# generate ten multiple imputations for 2x2 table, compute
# log-odds ratios and standard errors, and combine
fitML <- cvam( ~ V1 * V2, data=crime, freq=n ) # run EM first
set.seed(54981)
result <- cvam( fitML, method="MCMC",
   control=list( iterMCMC=5000, imputeEvery=500 ) )
impData <- get.imputedFreq(result)[-(1:2)] # just the frequencies
est.list <- std.err.list <- as.list(1:10)  # to hold the estimates and SEs
```

```
for( m in 1:10 ) {
   f <- impData[,m]
   est.list[[m]] <- log( (f[1] * f[4]) / (f[2] * f[3]) )
   std.err.list[[m]] <- sqrt( sum(1/f) )
}
miInference( est.list, std.err.list )
```

`seatbelt` *Seatbelt Data*

Description

This dataset, previously analyzed by Hochberg (1977), Chen (1989) and Schafer (1997), pertains to the effectiveness of seatbelts for preventing injury in automobile accidents. In a sample drawn from police reports, 80,084 accidents were classified by sex of the driver, degree of damage to the car (low or high), seatbelt use (no, yes) and whether the driver was injured (no, yes). Experience had shown that police tended to overestimate the proportion of drivers who were not using seatbelts and were not injured. To evaluate those potential biases, a followup study was conducted with 1,796 additional accidents. In the followup, investigators obtained more reliable information on seatbelt use and injury from personal interviews and hospital records.

Usage

```
seatbelt
```

Format

a data frame with 80 rows and 8 variables:

`source` source of the data; `"sample"` indicates the original sample of police reports, and `"followup"` indicates the followup study

`sex` sex of the driver (`"M"` or `"F"`)

`damage` degree of damage to the automobile (`"low"` or `"high"`)

`belt.p` whether the driver was wearing a seatbelt, according to the police report (`"no"` or `"yes"`)

`injury.p` whether the driver was injured, according to the police report (`"no"` or `"yes"`)

`belt.f` whether the driver was wearing a seatbelt, as determined in the followup study (`"no"` or `"yes"`)

`injury.f` whetherthe driver was injured, as determined in the followup study (`"no"` or `"yes"`)

Source

Chen, T.T. (1989) A review of methods for misclassified categorical data in epidemiology. *Statistics in Medicine*, 8, 1095-1106.

Hochberg, Y. (1977) On the use of double sampling schemes in analyzing categorical data with misclassification errors. *Journal of the American Statistical Association*, 72, 914-921.

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

summary.cvam *Summarize a* cvam *object*

Description

Summarizes the result from a call to cvam

Usage

```
## S3 method for class 'cvam'
summary(object, showCoef=TRUE, showEstimates=TRUE,
   digits = 4L, ...)
## S3 method for class 'cvam'
print(x, ...)
## S3 method for class 'summary.cvam'
print(x, ...)
```

Arguments

object	an object resulting from a call to cvam containing results from a fitted model.
showCoef	if TRUE, the table of coefficients will be displayed when the print method is invoked.
showEstimates	if TRUE, estimated marginal and conditional probabilities requested by the estimate argument will be displayed when the print method is invoked.
digits	for printing.
...	additional arguments to be passed to methods.
x	cvam or cvam.summary object

Value

for the summary method, an object of class "summary.cvam".

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

For more information, refer to the package vignette *Log-Linear Modeling with Missing and Coarsened Values Using the cvam Package*.

See Also

cvam

Examples

```
# saturated model for hivtest data
result <- cvam( ~ A*B*C*D, data=hivtest, freq=COUNT )
summary(result)
```

Index