Package ‘poLCAParallel’

March 13, 2026

Type Package

Title Polytomous Variable Latent Class Analysis Parallel

Version 1.2.7

Depends R (>= 4.1.2)

Imports Rcpp (>= 1.0.7), scatterplot3d, MASS, parallel, poLCA

LinkingTo Rcpp, RcppArmadillo (>= 0.12.4.1.0)

Description A 'C++' reimplementation of 'poLCA' - latent class analysis and latent class regression models for polytomous outcome variables, also known as latent structure analysis. It attempts to reproduce results and be as similar as possible to the original code, while running faster, especially with multiple repetitions, by utilising multiple threads. Further reading is available on the Queen Mary, University of London, IT Services Research blog <`https://blog.hpc.qmul.ac.uk/speeding_up_r_packages/`>.

URL `https://github.com/QMUL/poLCAParallel`

LazyLoad yes

NeedsCompilation yes

RoxygenNote 7.3.3

Encoding UTF-8

Suggests testthat (>= 3.0.0), roxygen2, usethis

Config/testthat/edition 3

SystemRequirements C++20

Author Sherman Lo [aut, cre],
Drew Linzer [aut],
Jeffrey Lewis. [aut]

Maintainer Sherman Lo <`s.lo@qmul.ac.uk`>

Repository CRAN

Date/Publication 2026-03-13 08:00:08 UTC

Contents

`blrt` *Bootstrap likelihood ratio test (non-regression only)*

Description

Does the bootstrap likelihood ratio test. Provide two fitted models, the null model and the alt model fitted with a different number of clusters/latent classes. Bootstrap samples are generated using the null model. The null model and the alt model are refitted on the bootstrap samples to investigate the log likelihood ratio of the two models.

Usage

```
blrt(
  model_null,
  model_alt,
  n_bootstrap,
  n_thread = parallel::detectCores(),
  n_rep = 1,
  max_iter = 1000,
  tol = 1e-10
)
```

Arguments

`model_null`	Fitted `poLCA` object, the null model
`model_alt`	Fitted `poLCA` object, the alt model
`n_bootstrap`	Number of bootstrap samples
`n_thread`	Number of threads
`n_rep`	Number of initial values to try when fitting on the bootstrap samples
`max_iter`	Maximum number of iterations for EM algorithm
`tol`	Tolerance for difference in log likelihood, used for stopping condition

Details

Runs in parallel for each bootstrap sample, potentially high memory if the data is large

Value

List containing the following:

- `fitted_log_ratio`: log likelihood ratio comparing the null and alt models
- `bootstrap_log_ratio`: vector of length `n_bootstrap`, bootstrapped log likelihood ratio comparing the null and alt models
- `p_value`: the proportion of bootstrap samples with log likelihood ratios greater than the fitted log likelihood ratio

---

`carcinoma` *Diagnoses of carcinoma (sample data)*

Description

Dichotomous ratings by seven pathologists of 118 slides for the presence or absence of carcinoma in the uterine cervix. Pathologists are labeled `A` through `G`. There were 20 different observed response patterns. This data set appears in Agresti (2002, p. 542) as Table 13.1.

Usage

```
data(carcinoma)
```

Format

A data frame with 118 observations on 7 variables representing pathologist ratings with 1 denoting "no" and 2 denoting "yes".

Source

Agresti, Alan. 2002. *Categorical Data Analysis, second edition*. Hoboken: John Wiley & Sons.

Examples

```
##
## Replication of latent class models in Agresti (2002, p. 543),
## Table 13.2 and Table 13.3.
##
data(carcinoma)
f <- cbind(A,B,C,D,E,F,G)~1
lca2 <- poLCA(f,carcinoma,nclass=2) # log-likelihood: -317.2568
lca3 <- poLCA(f,carcinoma,nclass=3) # log-likelihood: -293.705
lca4 <- poLCA(f,carcinoma,nclass=4,nrep=10,maxiter=5000) # log-likelihood: -289.2858
```

`cheating` *GPA and chronic cheating (sample data)*

Description

Dichotomous responses by 319 undergraduates to four questions about cheating behavior, and each student's academic GPA.

Students responded either (1) no or (2) yes as to whether they had ever lied to avoid taking an exam (`LIEEXAM`), lied to avoid handing a term paper in on time (`LIEPAPER`), purchased a term paper to hand in as their own or had obtained a copy of an exam prior to taking the exam (`FRAUD`), or copied answers during an exam from someone sitting near to them (`COPYEXAM`).

The `GPA` variable is partitioned into five groups: (1) 2.99 or less; (2) 3.00-3.25; (3) 3.26-3.50; (4) 3.51-3.75; (5) 3.76-4.00.

This data set appears in Dayton (1998, pp. 33 and 85) as Tables 3.4 and 7.1.

Usage

```
data(cheating)
```

Format

A data frame with 319 observations on 5 variables. Note: GPA data were not available for four students who reported never cheating.

Source

Dayton, C. Mitchell. 1998. *Latent Class Scaling Analysis*. Thousand Oaks, CA: SAGE Publications.

Examples

```
##
## Replication of latent class models in Dayton (1998)
##
## Example 1. Two-class LCA. (Table 3.3, p. 32)
##
```

```
data(cheating)
f <- cbind(LIEEXAM,LIEPAPER,FRAUD,COPYEXAM)~1
ch2 <- poLCA(f,cheating,nclass=2) # log-likelihood: -440.0271

##
## Example 2. Two-class latent class regression using
## GPA as a covariate to predict class membership as
## "cheaters" vs. "non-cheaters".
## (Table 7.1, p. 85, and Figure 7.1, p. 86)
##
f2 <- cbind(LIEEXAM,LIEPAPER,FRAUD,COPYEXAM)~GPA
ch2c <- poLCA(f2,cheating,nclass=2) # log-likelihood: -429.6384
GPAmat <- cbind(1,c(1:5))
exb <- exp(GPAmat %*% ch2c$coeff)
matplot(c(1:5),cbind(1/(1+exb),exb/(1+exb)),type="l",lwd=2,
        main="GPA as a predictor of persistent cheating",
        xlab="GPA category, low to high",
        ylab="Probability of latent class membership")
text(1.7,0.3,"Cheaters")
text(1.7,0.7,"Non-cheaters")

##
## Compare results from Example 1 to Example 2.
## Non-simultaneous estimation of effect of GPA on latent class
## membership biases the estimated effect in Example 1.
##
cheatcl <- which.min(ch2$P)
predcc <- sapply(c(1:5),function(v) mean(ch2$posterior[cheating$GPA==v,cheatcl],na.rm=TRUE))
## Having run Ex.2, add to plot:
matplot(c(1:5),cbind(1-predcc,predcc),type="l",lwd=2,add=TRUE)
text(4,0.14,"Cheaters\n (non-simul. estimate)")
text(4,0.87,"Non-cheaters\n (non-simul. estimate)")
```

`election` *2000 National Election Studies survey (sample data)*

Description

Survey data from the 2000 American National Election Study. Two sets of six questions with four responses each, asking respondents' opinions of how well various traits (moral, caring, knowledgable, good leader, dishonest, intelligent) describe presidential candidates Al Gore and George W. Bush. The responses are (1) Extremely well; (2) Quite well; (3) Not too well; (4) Not well at all. Many respondents have varying numbers of missing values on these variables.

The data set also includes potential covariates `VOTE3`, the respondent's 2000 vote choice (when asked); `AGE`, the respondent's age; `EDUC`, the respondent's level of education; `GENDER`, the respondent's gender; and `PARTY`, the respondent's Democratic-Republican partisan identification.

`VOTE3` is coded as (1) Gore; (2) Bush; (3) Other.

`EDUC` is coded as (1) 8 grades or less; (2) 9-11 grades, no further schooling; (3) High school diploma or equivalency; (4) More than 12 years of schooling, no higher degree; (5) Junior or community college level degree; (6) BA level degrees, no advanced degree; (7) Advanced degree.

`GENDER` is coded as (1) Male; (2) Female.

`PARTY` is coded as (1) Strong Democrat; (2) Weak Democrat; (3) Independent-Democrat; (4) Independent-Independent; (5) Independent-Republican; (6) Weak Republican; (7) Strong Republican.

Usage

```
data(election)
```

Format

A data frame with 1785 observations on 17 survey variables. Of these, 1311 individuals provided responses on all twelve candidate evaluations.

Source

The National Election Studies (https://www.electionstudies.org/). THE 2000 NATIONAL ELECTION STUDY [dataset]. Ann Arbor, MI: University of Michigan, Center for Political Studies [producer and distributor].

Examples

```
# Latent class models with one (loglinear independence) to three classes
data(election)
f <- cbind(MORALG,CARESG,KNOWG,LEADG,DISHONG,INTELG,
           MORALB,CARESB,KNOWB,LEADB,DISHONB,INTELB)~1
nes1 <- poLCA(f,election,nclass=1)  # log-likelihood: -18647.31
nes2 <- poLCA(f,election,nclass=2)  # log-likelihood: -17344.92
nes3 <- poLCA(f,election,nclass=3)  # log-likelihood: -16714.66

# Three-class model with a single covariate (party)
f2a <- cbind(MORALG,CARESG,KNOWG,LEADG,DISHONG,INTELG,
             MORALB,CARESB,KNOWB,LEADB,DISHONB,INTELB)~PARTY
nes2a <- poLCA(f2a,election,nclass=3,nrep=5)    # log-likelihood: -16222.32
pidmat <- cbind(1,c(1:7))
exb <- exp(pidmat %*% nes2a$coeff)
matplot(c(1:7),(cbind(1,exb)/(1+rowSums(exb))),ylim=c(0,1),type="l",
    main="Party ID as a predictor of candidate affinity class",
    xlab="Party ID: strong Democratic (1) to strong Republican (7)",
    ylab="Probability of latent class membership",lwd=2,col=1)
text(5.9,0.35,"Other")
text(5.4,0.7,"Bush affinity")
text(1.8,0.6,"Gore affinity")
```

---

gss82 *1982 General Social Survey (sample data)*

Description

Attitudes towards survey taking across two dichotomous and two trichotomous items among 1202 white respondents to the 1982 General Social Survey. Respondents give their opinion of the purpose of surveys (`PURPOSE`; good/depends/waste of time and money), the accuracy of surveys (`ACCURACY`; mostly true/not true), their understanding of survey questions (`UNDERSTA`; good/fair, poor), and how well they cooperated with the interviewer (`COOPERAT`; interested/cooperative/impatient, hostile). This data set appears in McCutcheon (1987, p. 30) as Table 3.1.

Usage

```
data(gss82)
```

Format

A data frame with 1202 observations on 4 survey variables.

Source

McCutcheon, A.L. 1987. *Latent class analysis*. Newbury Park: SAGE Publications.

Examples

```
data(gss82)
f <- cbind(PURPOSE,ACCURACY,UNDERSTA,COOPERAT)~1
gss.lc2 <- poLCA(f,gss82,nclass=2) # log-likelihood = -2783.268

# Could also try:
# gss.lc3 <- poLCA(f,gss82,nclass=3,maxiter=3000,nrep=10) # log-likelihood = -2754.545
# gss.lc4 <- poLCA(f,gss82,nclass=4,maxiter=15000,nrep=10,tol=1e-7) # log-likelihood = -2746.621
```

---

poLCA *Estimates latent class and latent class regression models for polytomous outcome variables*

Description

Latent class analysis, also known as latent structure analysis, is a technique for the analysis of clustering among observations in multi-way tables of qualitative/categorical variables. The central idea is to fit a model in which any confounding between the manifest variables can be explained by a single unobserved "latent" categorical variable. `poLCA` uses the assumption of local independence to estimate a mixture model of latent multi-way tables, the number of which (`nclass`) is specified by the user. Estimated parameters include the class-conditional response probabilities for each manifest variable, the "mixing" proportions denoting population share of observations corresponding to each latent multi-way table, and coefficients on any class-predictor covariates, if specified in the model.

Usage

```
poLCA(
  formula,
  data,
  nclass = 2,
  maxiter = 1000,
  graphs = FALSE,
  tol = 1e-10,
  na.rm = TRUE,
  probs.start = NULL,
  nrep = 1,
  verbose = TRUE,
  calc.se = TRUE,
  calc.chisq = TRUE,
  n.thread = parallel::detectCores(),
  se.smooth = FALSE
)
```

Arguments

`formula`	A formula expression of the form `response ~ predictors`. The details of model specification are given below.
`data`	A data frame containing variables in `formula`. Manifest variables must contain *only* integer values, and must be coded with consecutive values from 1 to the maximum number of outcomes for each variable. All missing values should be entered as `NA`.
`nclass`	The number of latent classes to assume in the model. Setting `nclass=1` results in `poLCA` estimating the loglinear independence model. The default is two.
`maxiter`	The maximum number of iterations through which the estimation algorithm will cycle.
`graphs`	Logical, for whether `poLCA` should graphically display the parameter estimates at the completion of the estimation algorithm. The default is `FALSE`.
`tol`	A tolerance value for judging when convergence has been reached. When the one-iteration change in the estimated log-likelihood is less than `tol`, the estimation algorithm stops updating and considers the maximum log-lparameterikelihood to have been found.
`na.rm`	Logical, for how `poLCA` handles cases with missing values on the manifest variables. If `TRUE`, those cases are removed (listwise deleted) before estimating the model. If `FALSE`, cases with missing values are retained. Cases with missing covariates are always removed. The default is `TRUE`.

`probs.start`	A list of matrices of class-conditional response probabilities to be used as the starting values for the estimation algorithm. Each matrix in the list corresponds to one manifest variable, with one row for each latent class, and one column for each outcome. The default is `NULL`, producing random starting values. Note that if `nrep>1`, then any user-specified `probs.start` values are only used in the first of the `nrep` attempts.
`nrep`	Number of times to estimate the model, using different values of `probs.start`. The default is one. Setting nrep>1 automates the search for the global—rather than just a local—maximum of the log-likelihood function. `poLCA` returns the parameter estimates corresponding to the model with the greatest log-likelihood.
`verbose`	Logical, indicating whether `poLCA` should output to the screen the results of the model. If `FALSE`, no output is produced. The default is `TRUE`.
`calc.se`	Logical, indicating whether `poLCA` should calculate the standard errors of the estimated class-conditional response probabilities and mixing proportions. The default is `TRUE`.
`calc.chisq`	Logical, indicating whether to calculate the goodness of fit statistics, the chi squared statistics and the log likelihood ratio. The default is `TRUE`.
`n.thread`	Integer, the number of threads to use. Each thread processes a repetition. By default, all detectable threads are used.
`se.smooth`	Logical, experimental, for calculating the standard errors, whether to smooth the outcome probabilities to produce more numerical stable results at the cost of bias.

Details

Model specification: Latent class models have more than one manifest variable, so the response variables are `cbind(dv1,dv2,dv3...)` where `dv#` refer to variable names in the data frame. For models with no covariates, the formula is `cbind(dv1,dv2,dv3)~1`. For models with covariates, replace the `~1` with the desired function of predictors `iv1,iv2,iv3...` as, for example, `cbind(dv1,dv2,dv3)~iv1+iv2*iv3`.

`poLCA` treats all manifest variables as qualitative/categorical/nominal – NOT as ordinal.

The implemention of this function in the package poLCAParallel is rewritten in C++. Multiple threads are used, where each thread processes an initial value or repetition.

Notes:

`poLCA` uses EM and Newton-Raphson algorithms to maximize the latent class model log-likelihood function. Depending on the starting parameters, this algorithm may only locate a local, rather than global, maximum. This becomes more and more of a problem as `nclass` increases. It is therefore highly advisable to run `poLCA` multiple times until you are relatively certain that you have located the global maximum log-likelihood. As long as `probs.start=NULL`, each function call will use different (random) initial starting parameters. Alternatively, setting `nrep` to a value greater than one enables the user to estimate the latent class model multiple times with a single call to `poLCA`, thus conducting the search for the global maximizer automatically.

The term "Latent class regression" (LCR) can have two meanings. In this package, LCR models refer to latent class models in which the probability of class membership is predicted by one or more covariates. However, in other contexts, LCR is also used to refer to regression models in which the manifest variable is partitioned into some specified number of latent classes as part of estimating the regression model. It is a way to simultaneously fit more than one regression to the data when the latent data partition is unknown. The `flexmix` function in package `flexmix` will estimate this other type of LCR model. Because of these terminology issues, the LCR models this package estimates are sometimes termed "latent class models with covariates" or "concomitant-variable latent class analysis," both of which are accurate descriptions of this model.

The package poLCAParallel reimplements the poLCA fitting, standard error calculations, goodness of fit tests and the bootstrap log-likelihood ratio test in C++. This was done using Rcpp and RcppArmadillo which allows R to run fast C++ code. Additional notes include:

- The API remains the same as the original poLCA with a few additions
- It tries to reproduce results from the original poLCA
- The code uses Armadillo for linear algebra
- Multiple repetitions are done in parallel using `std::jthread` for multi-thread programming and `std::mutex` to prevent data races
- Direct inversion of matrices is avoided to improve numerical stability and performance
- Response probabilities are reordered to increase cache efficiency
- Use of `std::map` for the chi-squared calculations to improve performance

Further reading is available on the QMUL ITS Research Blog.

References:

- Agresti, Alan. 2002. *Categorical Data Analysis, second edition*. Hoboken: John Wiley & Sons.
- Bandeen-Roche, Karen, Diana L. Miglioretti, Scott L. Zeger, and Paul J. Rathouz. 1997. "Latent Variable Regression for Multiple Discrete Outcomes." *Journal of the American Statistical Association*. 92(440): 1375-1386.
- Hagenaars, Jacques A. and Allan L. McCutcheon, eds. 2002. *Applied Latent Class Analysis*. Cambridge: Cambridge University Press.
- McLachlan, Geoffrey J. and Thriyambakam Krishnan. 1997. *The EM Algorithm and Extensions*. New York: John Wiley & Sons.

Value

an object of class poLCA; a list containing the following elements:

- `y`: data frame of manifest variables.
- `x`: data frame of covariates, if specified.
- `N`: number of cases used in model.
- `Nobs`: number of fully observed cases (less than or equal to `N`).
- `probs`: estimated class-conditional response probabilities.
- `probs.se`: standard errors of estimated class-conditional response probabilities, in the same format as `probs`.
- `P`: sizes of each latent class; equal to the mixing proportions in the function basic latent class model, or the mean of the priors in the latent class regression model.
- `P.se`: the standard errors of the estimated `P`.

- `prior`: matrix of prior class membership probabilities
- `posterior`: matrix of posterior class membership probabilities; also see function `poLCA.posterior`.
- `predclass`: vector of predicted class memberships, by modal assignment.
- `predcell`: table of observed versus predicted cell counts for cases with no missing values; also see functions `poLCA.table` and `poLCA.predcell`
- `llik`: maximum value of the log-likelihood.
- `numiter`: number of iterations until reaching convergence.
- `maxiter`: maximum number of iterations through which the estimation algorithm was set to run.
- `coeff`: multinomial logit coefficient estimates on covariates (when estimated). `coeff` is a matrix with `nclass-1` columns, and one row for each covariate. All logit coefficients are calculated for classes with respect to class 1.
- `coeff.se`: standard errors of coefficient estimates on covariates (when estimated), in the same format as `coeff`.
- `coeff.V`: covariance matrix of coefficient estimates on covariates (when estimated).
- `aic`: Akaike Information Criterion.
- `bic`: Bayesian Information Criterion.
- `Gsq`: Likelihood ratio/deviance statistic.
- `Chisq`: Pearson Chi-square goodness of fit statistic for fitted vs. observed multiway tables.
- `time`: length of time it took to run the model.
- `npar`: number of degrees of freedom used by the model (estimated parameters).
- `resid.df`: number of residual degrees of freedom.
- `attempts`: a vector containing the maximum log-likelihood values found in each of the `nrep` attempts to fit the model.
- `eflag`: Logical, error flag. `TRUE` if estimation algorithm needed to automatically restart with new initial parameters. A restart is caused in the event of computational/rounding errors that result in nonsensical parameter estimates.
- `probs.start`: A list of matrices containing the class-conditional response probabilities used as starting values in the estimation algorithm. If the algorithm needed to restart (see `eflag`), then this contains the starting values used for the final, successful, run.
- `probs.start.ok`: Logical. `FALSE` if `probs.start` was incorrectly specified by the user, otherwise `TRUE`.
- `call`: function call to `poLCA`.

Examples

```
##
## Three models without covariates:
## M0: Loglinear independence model.
## M1: Two-class latent class model.
## M2: Three-class latent class model.
##
data(values)
```

```
f <- cbind(A, B, C, D)~1
M0 <- poLCA(f, values, nclass = 1) # log-likelihood: -543.6498
M1 <- poLCA(f, values, nclass = 2) # log-likelihood: -504.4677
# log-likelihood: -503.3011
M2 <- poLCA(f, values, nclass = 3, maxiter = 8000)

##
## Three-class model with a single covariate.
##
data(election)
f2a <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
  MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
)~PARTY
# log-likelihood: -16222.32
nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5)
pidmat <- cbind(1, c(1:7))
exb <- exp(pidmat %*% nes2a$coeff)
matplot(c(1:7), (cbind(1, exb) / (1 + rowSums(exb))),
  ylim = c(0, 1), type = "l",
  main = "Party ID as a predictor of candidate affinity class",
  xlab = "Party ID: strong Democratic (1) to strong Republican (7)",
  ylab = "Probability of latent class membership", lwd = 2, col = 1
)
text(5.9, 0.35, "Other")
text(5.4, 0.7, "Bush affinity")
text(1.8, 0.6, "Gore affinity")
```

`poLCA.entropy` *Entropy of a fitted latent class model*

Description

Calculates the entropy of a cross-classification table produced as a density estimate using a latent class model.

Usage

```
poLCA.entropy(lc)
```

Arguments

`lc`	A model object estimated using the `poLCA` function.

Details

Entropy is a measure of dispersion (or concentration) in a probability mass function. For multivariate categorical data it is calculated

$$H = -\sum_c p_c log(p_c)$$

where p_c is the share of the probability in the cth cell of the cross-classification table. A fitted latent class model produces a smoothed density estimate of the underlying distribution of cell percentages in the multi-way table of the manifest variables. This function calculates the entropy of that estimated probability mass function.

Value

A number taking a minumum value of 0 (representing complete concentration of probability on one cell) and a maximum value equal to the logarithm of the total number of cells in the fitted cross-classfication table (representing complete dispersion, or equal probability for outcomes across every cell).

See Also

`poLCA`

Examples

```
data(carcinoma)
f <- cbind(A, B, C, D, E, F, G) ~ 1
lca2 <- poLCA(f, carcinoma, nclass = 2) # log-likelihood: -317.2568
lca3 <- poLCA(f, carcinoma, nclass = 3) # log-likelihood: -293.705
# log-likelihood: -289.2858
lca4 <- poLCA(f, carcinoma, nclass = 4, nrep = 10, maxiter = 5000)

# Maximum entropy (if all cases equally dispersed)
log(prod(sapply(lca2$probs, ncol)))

# Sample entropy ("plug-in" estimator, or MLE)
p.hat <- lca2$predcell$observed / lca2$N
H.hat <- -sum(p.hat * log(p.hat))
H.hat # 2.42

# Entropy of fitted latent class models
poLCA.entropy(lca2)
poLCA.entropy(lca3)
poLCA.entropy(lca4)
```

`poLCA.posterior` *Posterior probabilities from a latent class model*

Description

Calculates the posterior probability that cases belong to each latent class.

Usage

```
poLCA.posterior(lc, y, x = NULL)
```

Arguments

`lc`	A model object estimated using the `poLCA` function.
`y`	A vector or matrix containing series of responses on the manifest variables in `lc`.
`x`	An optional vector or matrix of covariate values, if `lc` was specified as a latent class regression model.

Details

From the parameters estimated by the latent class model, this function calculates the "posterior" probability that a specified case – characterized by values of the manifest variables `y`, and, if a latent class regression model, concomitant variables `x` – "belongs to" each latent class in `lc`. For observed cases, this information is also contained in the `lc` model object as `lc$posterior`. The added benefit of this function is that it can calculate posterior class membership probabilities for arbitrary values of `x` and `y`, whether or observed or not.

Value

A matrix containing posterior probabilities corresponding to the specified sets of responses `y`, based on the estimated latent class model `lc`. For each row (one case), the first column gives the posterior probability of being in class 1, the second column gives the posterior probability of being in class 2, and so forth. Across rows, these probabilities sum to one.

See Also

`poLCA`

Examples

```
data(election)

## Basic latent class model with three classes
f1 <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
  MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
) ~ 1
lc1 <- poLCA(f1, election, nclass = 3) # log-likelihood: -16714.66

# The first observed case
lc1$y[1, ]
lc1$posterior[1, ]
poLCA.posterior(lc = lc1, y = as.numeric(lc1$y[1, ]))

# A hypothetical case
poLCA.posterior(lc = lc1, y = rep(2, 12))

# Entering y as a matrix
lc1$posterior[1:10, ]
poLCA.posterior(lc = lc1, y = mapply(as.numeric, lc1$y[1:10, ]))
```

```
## Latent class regression model with three classes
f2 <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG, MORALB, CARESB,
  KNOWB, LEADB, DISHONB, INTELB
) ~ AGE + EDUC + GENDER
lc2 <- poLCA(f2, election, nclass = 3) # log-likelihood: -16598.38

# Posteriors for case number 97 (poorly classified)
lc2$y[97, ]
lc2$x[97, ]
lc2$posterior[97, ]
poLCA.posterior(lc = lc2, y = as.numeric(lc2$y[97, ]), x = c(41, 6, 1))

# If x is not specified, the posterior is calculated using the population
# average
poLCA.posterior(lc = lc2, y = as.numeric(lc2$y[97, ]))

# Entering y and x as matrices
round(lc2$posterior[95:100, ], 2)
round(poLCA.posterior(
  lc = lc2, y = mapply(as.numeric, lc2$y[95:100, ]),
  x = as.matrix(lc2$x[95:100, -1])
), 2)
```

`poLCA.predcell` *Predicted cell percentages in a latent class model*

Description

Calculates the predicted cell percentages from a latent class model, for specified values of the manifest variables.

Usage

```
poLCA.predcell(lc, y)
```

Arguments

`lc`	A model object estimated using the `poLCA` function.
`y`	A vector or matrix containing series of responses on the manifest variables in `lc`.

Details

The parameters estimated by a latent class model can be used to produce a density estimate of the underlying probability mass function across the cells in the multi-way table of manifest variables. This function calculates cell percentages for that density estimate, corresponding to selected sets of responses on the manifest variables, `y`.

Value

A vector containing cell percentages corresponding to the specified sets of responses `y`, based on the estimated latent class model `lc`.

See Also

`poLCA`

Examples

```
data(carcinoma)
f <- cbind(A, B, C, D, E, F, G) ~ 1
lca3 <- poLCA(f, carcinoma, nclass = 3) # log-likelihood: -293.705

# Only 20 out of 32 possible response patterns are observed
lca3$predcell

# Produce cell probabilities for one sequence of responses
poLCA.predcell(lc = lca3, y = c(1, 1, 1, 1, 1, 1, 1))

# Estimated probabilities for a cell with zero observations
poLCA.predcell(lc = lca3, y = c(1, 1, 1, 1, 1, 1, 2))

# Cell probabilities for both cells at once; y entered as a matrix
poLCA.predcell(lc = lca3, y = rbind(
  c(1, 1, 1, 1, 1, 1, 1),
  c(1, 1, 1, 1, 1, 1, 2)
))
```

`poLCA.reorder` *Reorder latent classes in poLCA*

Description

A helper function to simplify the reordering of latent classes returned by `poLCA`

Usage

```
poLCA.reorder(probs, o.new)
```

Arguments

`probs`	a list of class-conditional response probabilities previously used as start values to estimate a particular latent class model using `poLCA`.
`o.new`	a vector of length equal to the number of latent classes in `probs`, giving the desired reordering of the latent classes.

Details

Because the latent classes outputted by `poLCA` are unordered categories, the numerical order of the classes is arbitrary, and is determined solely by the initial values of the EM algorithm. If `probs.start` is set to `NULL` (the default) when calling `poLCA`, then the function generates the starting values randomly in each run, typically rearranging the latent class labels. The `poLCA.reorder` function is a convenient way to manually adjust the order of the latent classes, by changing the order of the `probs.start`. Refitting the latent class model using these reordered start values will produce a model having the desired category labels.

Value

A list of matrices containing the rearranged (by row) class-conditional response probabilities.

See Also

`poLCA`

Examples

```
##
## Using the "cheating" sample data set, make the larger
## non-cheater class the first ("reference") class in a
## latent class regression model. The coefficient on GPA
## now maintains a consistent interpretation.
##
data(cheating)
f2 <- cbind(LIEEXAM, LIEPAPER, FRAUD, COPYEXAM) ~ GPA
lc.ch <- poLCA(f2, cheating, nclass = 2, verbose = FALSE)
probs.start.new <- poLCA.reorder(
  lc.ch$probs.start,
  order(lc.ch$P, decreasing = TRUE)
)
lc.ch <- poLCA(f2, cheating, nclass = 2, probs.start = probs.start.new)
```

`poLCA.simdata` *Create simulated cross-classification data*

Description

Uses the latent class model's assumed data-generating process to create a simulated dataset that can be used to test the properties of the poLCA latent class and latent class regression estimator.

Usage

```
poLCA.simdata(
  N = 5000,
  probs = NULL,
  nclass = 2,
  ndv = 4,
  nresp = NULL,
  x = NULL,
  niv = 0,
  b = NULL,
  P = NULL,
  missval = FALSE,
  pctmiss = NULL
)
```

Arguments

`N`	number of observations.
`probs`	a list of matrices of dimension `nclass` by `nresp` with each matrix corresponding to one manifest variable, and each row containing the class-conditional outcome probabilities (which must sum to 1). If `probs` is `NULL` (default) then the outcome probabilities are generated randomly.
`nclass`	number of latent classes. If `probs` is specified, then `nclass` is set equal to the number of rows in each matrix in that list. If `P` is specified, then `nclass` is set equal to the length of that vector. If `b` is specified, then `nclass` is set equal to one greater than the number of columns in `b`. Otherwise, the default is two.
`ndv`	number of manifest variables. If `probs` is specified, then `ndv` is set equal to the number of matrices in that list. If `nresp` is specified, then `ndv` is set equal to the length of that vector. Otherwise, the default is four.
`nresp`	number of possible outcomes for each manifest variable. If `probs` is specified, then `ndv` is set equal to the number of columns in each matrix in that list. If both `probs` and `nresp` are `NULL` (default), then the manifest variables are assigned a random number of outcomes between two and five.
`x`	a matrix of concomicant variables with `N` rows and `niv` columns. If `x=NULL` (default), but `niv>0`, then `niv` concomitant variables will be generated as mutually independent random draws from a standard normal distribution.
`niv`	number of concomitant variables (covariates). Setting `niv=0` (default) creates a data set assuming no covariates. If `nclass=1` then `niv` is automatically set equal to 0. If both `x` and `niv` are entered, then the number of columns in `x` overrides the value of `niv`. The number of rows in `b`, less one, also overrides `niv`.
`b`	when using covariates, an `niv+1` by `nclass-1` matrix of (multinomial) logit coefficients. If `b` is `NULL` (default), then coefficients are generated as random integers between -2 and 2.
`P`	a vector of mixing proportions (class population shares) of length `nclass`. `P` must sum to 1. Disregarded if `b` is specified or `niv>1` because then `P` is, in part, a function of the concomitant variables. If `P` is `NULL` (default), then the mixing proportions are generated randomly.

`missval`	logical. If `TRUE` then a fraction `pctmiss` of the manifest variables are randomly dropped as missing values. Default is `FALSE`.
`pctmiss`	percentage of values to be dropped as missing, if `missval=TRUE`. If `pctmiss` is `NULL` (default), then a value between 5 and 40 percent is chosen randomly.

Details

Note that entering `probs` overrides `nclass`, `ndv`, and `nresp`. It also overrides `P` if the length of the `P` vector is not equal to the length of the `probs` list. Likewise, if `probs=NULL`, then `length(nresp)` overrides `ndv` and `length(P)` overrides `nclass`. Setting `niv>1` causes any user-entered value of `P` to be disregarded.

Value

A list containing the following

- `dat`: a data frame containing the simulated variables. Variable names for manifest variables are Y1, Y2, etc. Variable names for concomitant variables are X1, X2, etc.
- `probs`: a list of matrices of dimension `nclass` by `nresp` containing the class-conditional response probabilities.
- `nresp`: a vector containing the number of possible outcomes for each manifest variable.
- `b`: coefficients on covariates, if used.
- `P`: mixing proportions corresponding to each latent class.
- `pctmiss`: percent of observations missing.
- `trueclass`: `N` by 1 vector containing the "true" class membership for each individual.

See Also

`poLCA`

Examples

```
# Create a sample data set with 3 classes and no covariates
# and run poLCA to recover the specified parameters.
probs <- list(
  matrix(c(0.6, 0.1, 0.3, 0.6, 0.3, 0.1, 0.3, 0.1, 0.6),
    ncol = 3, byrow = TRUE
  ), # conditional resp prob to Y1
  matrix(c(0.2, 0.8, 0.7, 0.3, 0.3, 0.7),
    ncol = 2, byrow = TRUE
  ), # conditional resp prob to Y2
  matrix(c(0.3, 0.6, 0.1, 0.1, 0.3, 0.6, 0.3, 0.6, 0.1),
    ncol = 3, byrow = TRUE
  ), # conditional resp prob to Y3
  matrix(c(0.1, 0.1, 0.5, 0.3, 0.5, 0.3, 0.1, 0.1, 0.3, 0.1, 0.1, 0.5),
    ncol = 4, byrow = TRUE
  ), # conditional resp prob to Y4
  matrix(c(0.1, 0.1, 0.8, 0.1, 0.8, 0.1, 0.8, 0.1, 0.1),
    ncol = 3, byrow = TRUE
```

```
  )
) # conditional resp prob to Y5
simdat <- poLCA.simdata(N = 1000, probs, P = c(0.2, 0.3, 0.5))
f1 <- cbind(Y1, Y2, Y3, Y4, Y5) ~ 1
lc1 <- poLCA(f1, simdat$dat, nclass = 3)
table(lc1$predclass, simdat$trueclass)

# Create a sample dataset with 2 classes and three covariates.
# Then compare predicted class memberships when the model is
# estimated "correctly" with covariates to when it is estimated
# "incorrectly" without covariates.
simdat2 <- poLCA.simdata(
  N = 1000, ndv = 7, niv = 3, nclass = 2,
  b = matrix(c(1, -2, 1, -1))
)
f2a <- cbind(Y1, Y2, Y3, Y4, Y5, Y6, Y7) ~ X1 + X2 + X3
lc2a <- poLCA(f2a, simdat2$dat, nclass = 2)
f2b <- cbind(Y1, Y2, Y3, Y4, Y5, Y6, Y7) ~ 1
lc2b <- poLCA(f2b, simdat2$dat, nclass = 2)
table(lc2a$predclass, lc2b$predclass)
```

`poLCA.table` *Frequency tables of predicted cell counts from latent class analysis*

Description

Calculates predicted cell frequencies based on an estimated latent class model.

Usage

```
poLCA.table(formula, condition = NULL, lc)
```

Arguments

`formula`	A formula expression of the form `variable ~ 1` for a one-way frequency distribution, or `row ~ column` for two way-tables.
`condition`	A list containing the values of the manifest variables to hold fixed when creating the table specified by the `formula` argument. Setting this to an empty list, `condition=list()`, conditions on none of the other manifest variables, producing the marginal frequencies.
`lc`	A model object previously estimated using the `poLCA` function.

Details

This function outputs predicted cell counts for user-specified combinations of the manifest variables, based on a latent class model estimated by the `poLCA` function. The `predcell` table outputted automatically by `poLCA` also contains predicted cell frequencies, but only for cells containing at least one observation. In contrast, `poLCA.table` will calculate predicted cell counts for all cells, including those with zero observations.

Value

A vector or table containing the specified frequency distribution.

See Also

`poLCA`

Examples

```
data(gss82)
f <- cbind(PURPOSE, ACCURACY, UNDERSTA, COOPERAT) ~ 1
gss.lc2 <- poLCA(f, gss82, nclass = 2)
gss.lc2$predcell

poLCA.table(
  formula = COOPERAT ~ 1,
  condition = list(PURPOSE = 3, ACCURACY = 1, UNDERSTA = 2),
  lc = gss.lc2
)
poLCA.table(
  formula = COOPERAT ~ UNDERSTA,
  condition = list(PURPOSE = 3, ACCURACY = 1),
  lc = gss.lc2
)
poLCA.table(
  formula = COOPERAT ~ UNDERSTA,
  condition = list(),
  lc = gss.lc2
)
```

`poLCAParallel` *poLCAParallel: Parallelized Polytomous Variable Latent Class Analysis*

Description

The package poLCAParallel reimplements poLCA fitting, standard error calculations, goodness of fit tests and the bootstrap log-likelihood ratio test in C++.

`poLCAParallel.goodnessfit` *Add predcell, Gsq and Chisq to a fitted poLCA object*

Description

Finds unique responses and put them in a dataframe along with the observed and expected frequencies. Also calculates the log likelihood ratio and chi squared statistics.

Usage

```
poLCAParallel.goodnessfit(results)
```

Arguments

`results`	A model object estimated using the `poLCA` function

Details

Wrapper function around the C++ function `GoodnessFitRcpp()`.

Value

The model object with three added items:

- `predcell`: dataframe of unique responses with their observed and expected frequencies
- `Gsq`: log likelihood ratio
- `Chisq`: chi squared statistic

`poLCAParallel.se` *Calculates the standard errors and adds them to a fitted poLCA model object*

Description

Calculates the standard errors and adds them, as attributes, to a fitted poLCA model object `$P.se`, `$probs.se`, `$coeff.se` and `$coeff.V`

Usage

```
poLCAParallel.se(polca, is_smooth = FALSE)
```

Arguments

`polca`	A model object estimated using the `poLCA` function
`is_smooth`	Logical, experimental, for calculating the standard errors, whether to smooth the outcome probabilities to produce more numerical stable results at the cost of bias.

Details

R wrapper function for the C++ function `StandardErrorRcpp()`
`$coeff.se` and `$coeff.V` are set to `NA` if the provided poLCA is a poLCA problem with no regression

Value

the fitted poLCA model object with the attributes `$P.se`, `$probs.se`, `$coeff.se` and '$coeff.V' modified or added

---

`rmulti` *Random draws from a multinomial distribution*

Description

One random draw from a multinomial distribution or list of multinomial distributions.

Usage

```
rmulti(p)
```

Arguments

`p`	matrix of dimension `n` by `r` containing probabilities, for each row, of drawing each of `r` outcomes. `p` may also be entered as a vector, in which case `rmulti` treats it as a matrix of dimension `n=1` by `r`. Each row of matrix `p` must sum to 1 or `rmulti` will not work properly.

Value

A vector of length `n`. Each item represents one draw from the multinomial distribution parameterized by the outcome probabilities in each row of `p`.

Examples

```
##
## One draw from a three-category multinomial distribution.
##
p1 <- c(0.7, 0.2, 0.1)
rmulti(p1)

##
## 10,000 draws from a three-category multinomial distribution.
##
n <- 10000
p2 <- matrix(p1, nrow = n, ncol = length(p1), byrow = TRUE)
rmdraws <- rmulti(p2)
table(rmdraws) / n # should be approximately 0.7, 0.2, 0.1

##
## 10,000 draws from a mixture of three groups of a
## four-category multinomial distribution.
##
group.p <- matrix(c(0.5, 0.3, 0.2), nrow = n, ncol = 3, byrow = TRUE)
group <- rmulti(group.p)
p3 <- t(matrix(NA, nrow = n, ncol = 4))
p3[, group == 1] <- c(0.7, 0.1, 0.1, 0.1)
p3[, group == 2] <- c(0.1, 0.7, 0.1, 0.1)
p3[, group == 3] <- c(0.1, 0.1, 0.1, 0.7)
p3 <- t(p3)
rmdraws3 <- rmulti(p3)
table(group, rmdraws3)
table(group, rmdraws3) / rowSums(table(group, rmdraws3))
```

`values` *Universalistic vs. particularistic values (sample data)*

Description

Dichotomous survey responses from 216 respondents to four questions (`A`, `B`, `C`, `D`) measuring tendencies towards "universalistic" or "particularistic" values. This data set appears in Goodman (2002, p. 14) as Table 4, and previously appeared in Goodman (1974) and Stouffer and Toby (1951).

Usage

```
data(values)
```

Format

A data frame with 216 observations on 4 variables representing survey responses to dichotomous questions, with 1 denoting the "particularistic" values response and 2 denoting the "universalistic" values response.

Source

Stouffer, S.A. and J. Toby. 1951. "Role conflict and personality." *American Journal of Sociology*. 56: 395:406.

Goodman, Leo A. 1974. "Exploratory Latent-Structure Analysis Using Both Identifiable and Unidentifiable Models." *Biometrika*. 61(2): 215-231.

Goodman, Leo A. 2002. "Latent Class Analysis; The Empirical Study of Latent Types, Latent Variables, and Latent Structures." in Jacques A. Hagenaars and Allan L. McCutcheon, eds. *Applied Latent Class Analysis*. Cambridge: Cambridge University Press.

Examples

```
##
## Replication of latent class models in Goodman (2002),
## Tables 5b, 5c, and 6.
##
data(values)
f <- cbind(A,B,C,D)~1
M0 <- poLCA(f,values,nclass=1) # log-likelihood: -543.6498
M1 <- poLCA(f,values,nclass=2) # log-likelihood: -504.4677
M2 <- poLCA(f,values,nclass=3,maxiter=8000) # log-likelihood: -503.3011
```

Index