Package ‘covLCA’

February 19, 2015

Type Package
Title Latent Class Models with Covariate Effects on Underlying and Measured Variables
Version 1.0
Date 2013-05-03
Author Aurelie Bertrand <aurelie.bertrand@uclouvain.be> and Christian M. Hafner <christian.hafner@uclouvain.be>
Maintainer Aurelie Bertrand <aurelie.bertrand@uclouvain.be>
Description Estimation of latent class models with covariate effects on underlying and measured variables. The measured variables are dichotomous or polytomous, all with the same number of categories.
License GPL-2
Depends Matrix, mlogit, poLCA
Repository CRAN
NeedsCompilation no
Date/Publication 2013-06-10 11:06:36

R topics documented:

covLCA ................................................................. 1

Index 8

---

covLCA  Latent Class Models with Covariate Effects on Underlying and Measured Variables

Description

Fits latent class models with covariate effects on underlying and measured variables. The measured variables are dichotomous or polytomous, all with the same number of categories.
Usage

covLCA(formula1, formula2, data, nclass = 2, maxiter = 1000, tol = 1e-10,
beta.start = NULL, alpha.start = NULL, gamma.start = NULL, beta.auto = TRUE,
alpha.auto = TRUE, gamma.auto = TRUE, nrep = 1, verbose = TRUE, calc.se = TRUE)

Arguments

formula1 The formula where the dependent variables are the manifest variables, grouped
by `cbind()`, and the independent variables are the covariates for the latent class
probabilities.

formula2 The formula where the dependent variables are the manifest variables, grouped
by `cbind()`, and the independent variables are the covariates for the conditional
probabilities.

data a dataframe containing all variables appearing in `formula1` and `formula2`. Man-
ifest variables must contain only integer values, and must be coded with consec-
utive values from 1 to the maximum number of outcomes for each variable. All
missing values should be entered as NA and all cases containing missing values
(in the manifest variables or in the covariates) are removed before estimating the
model.

nclass the number of latent classes assumed in the model.

maxiter the maximum number of iterations through which the estimation algorithm will
cycle.

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 esti-
mation algorithm stops updating and considers the maximum log-likelihood to
have been found.

beta.start a vector of parameters $\beta_{jp}$ to be used as the starting values for the estimation
algorithm. There is one parameter for each pair latent class-covariate (the index
of the covariate moving faster), except the last class, considered as the reference,
for which $\beta_{jp} = 0 \forall p$. The default is NULL, leading either to an automatic search
for “reasonable” initial values (when `beta.auto=TRUE`, the default) or to the
generation of random starting values (when `beta.auto=FALSE`). Note that if
`nrep>1`, then any user-specified `beta.start` values are only used in the first
of the `nrep` attempts.

alpha.start an $M \times L(K - 1)$ matrix of parameters $\alpha_{mlk}$ to be used as the starting values
for the estimation algorithm. Rows correspond to manifest variable $m$. Within
each row, columns correspond to covariates $l$ and categories of manifest vari-
ables $k$ (except the last category, for which $\alpha_{mlK,m} = 0$), the index of the latter
moving faster. The default is NULL, leading either to an automatic search for
“reasonable” initial values (when `alpha.auto=TRUE`, the default) or to the
generation of random starting values (when `alpha.auto=FALSE`). Note that if `nrep
> 1`, then any user-specified `alpha.start` values are only used in the first of
the `nrep` attempts.

gamma.start an $M \times J(K - 1)$ matrix of parameters $\gamma_{mjk}$ to be used as the starting values
for the estimation algorithm. Rows correspond to manifest variable $m$. Within
each row, columns correspond to latent classes $j$ and categories of manifest
variables $k$ (except the last category, for which $\gamma_{mjk}\omega = 0$), the index of the latter moving faster. The default is NULL, leading either to an automatic search for “reasonable” initial values (when beta.auto=TRUE, the default) or to the generation of random starting values (when beta.auto=FALSE). Note that if nrep > 1, then any user-specified gamma.start values are only used in the first of the nrep attempts.

**beta.auto** logical, indicating whether covLCA() should calculate “reasonable” initial values for parameters $\beta$. If TRUE, the approach advised by Huang and Bandeen-Roche (2004) is applied: a standard latent class model assuming nclass latent classes is estimated, then each individual is assigned to a class with the posterior probabilities of class membership from this model, and finally a multinomial logistic regression model relating the latent classes to covariates $x$ is fitted, whose coefficient estimates give initial estimates of $\beta$. If FALSE, either random initial values are generated (if beta.start=NULL) or values provided by the user are used.

**alpha.auto** logical, indicating whether covLCA() should calculate “reasonable” initial values for parameters $\alpha$. If TRUE, the approach advised by Huang and Bandeen-Roche (2004) is applied: $M$ different multinomial logistic regression models for $(Y_{i1}, z_{i1}), \ldots, (Y_{iM}, z_{iM})$ are fitted and the corresponding estimated coefficients are initial values for parameters $\alpha$. If FALSE, either random initial values are generated (if alpha.start=NULL) or values provided by the user are used.

**gamma.auto** logical, indicating whether covLCA() should calculate “reasonable” initial values for parameters $\gamma$. If TRUE, the approach advised by Huang and Bandeen-Roche (2004) is applied: $M$ different multinomial logistic regression models for $(Y_{i1}, z_{i1}), \ldots, (Y_{iM}, z_{iM})$ are fitted and the corresponding estimated coefficients are initial values for parameters $\gamma$. If FALSE, either random initial values are generated (if gamma.start=NULL) or values provided by the user are used.

**nrep** number of times the model is estimated, using different values of beta.start, alpha.start and gamma.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. covLCA() returns the parameter estimates corresponding to the model with the greatest log-likelihood.

**verbose** logical, indicating whether covLCA() should output to the screen the results of the model.

**calc.se** logical, indicating whether covLCA() should calculate the standard errors of the estimated parameters $\beta_{jp}$, $\alpha_{mk}$ and $\gamma_{mjk}$.

### Details

We denote individuals by $i$ ($i=1, \ldots, N$), manifest variables (items) by $Y_m$ ($m=1, \ldots, M$), levels of the manifest variables by $k$ ($k=1, \ldots, K$), the latent variable by $S$ ($S=j$ and $j=1, \ldots, J$). There are two sets of covariates: those related with the latent class probabilities, $x_i = (1, x_{i1}, \ldots, x_{ip})^T$, and those which can have a direct effect on the manifest variables, $z_i = (z_{i1}, \ldots, z_{iM})$ with $z_{im} = (1, z_{im1}, \ldots, z_{imL})^T$, $m=1, \ldots, M$. The parameters of the model are the latent class probabilities $\pi_j(x_i | \beta) = P(S_i = j; x_i)$ and the conditional probabilities $p_{mkj}(\gamma_{mj} + z'_{im} \alpha_m) = P(Y_{im} = k | S_i = j; z_{im})$.

The model is
\[ P(Y_i = y | x_i, z_i) = P(Y_{i1} = y_1, \ldots, Y_{iM} = y_M | x_i, z_i) \]
\[ = \sum_{j=1}^{J} \left\{ \pi_j(x'_i \beta) \prod_{m=1}^{M} \pi^{m_{mkj}}(\gamma_{mj} + z'_{im} \alpha_m) \right\} \]

with
\[ \log \left( \frac{\pi_j(x'_i \beta)}{\pi_j(x'_i \beta)} \right) = x'_i \beta_j \quad i = 1, \ldots, N; \quad j = 1, \ldots, (J - 1) \]

and
\[ \log \left( \frac{p_{mkj}(\gamma_{mj} + z'_{im} \alpha_m)}{p_{mKj}(\gamma_{mj} + z'_{im} \alpha_m)} \right) = \gamma_{mkj} + z'_{im} \alpha_{mk} \]

**Value**

The output of function `covLCA()` is a list containing the following elements:

- **llik** The log-likelihood value of the estimated model.
- **attempts** A vector containing the maximum loglikelihood values found in each of the `nrep` attempts to fit the model.
- **beta.start** A vector containing the initial values for parameters $\beta$ when such values were provided by the user (in `beta.start`) or when they were randomly generated (when `beta.start=NULL` and `beta.auto=FALSE`).
- **alpha.start** A vector containing the initial values for parameters $\alpha$ when such values were provided by the user (in `alpha.start`) or when they were randomly generated (when `alpha.start=NULL` and `alpha.auto=FALSE`).
- **gamma.start** A vector containing the initial values for parameters $\gamma$ when such values were provided by the user (in `gamma.start`) or when they were randomly generated (when `gamma.start=NULL` and `gamma.auto=FALSE`).
- **beta.auto** Logical, indicating whether the user asked for “reasonable” initial estimates of parameters $\beta$ to be automatically computed (with the argument `beta.auto`).
- **alpha.auto** Logical, indicating whether the user asked for “reasonable” initial estimates of parameters $\alpha$ to be automatically computed (with the argument `alpha.auto`).
- **gamma.auto** Logical, indicating whether the user asked for “reasonable” initial estimates of parameters $\gamma$ to be automatically computed (with the argument `gamma.auto`).
- **beta.initAuto** A vector containing the initial values for parameters $\beta$ when “reasonable” values are automatically computed (when `beta.auto=TRUE`).
- **alpha.initAuto** A vector containing the initial values for parameters $\alpha$ when “reasonable” values are automatically computed (when `alpha.auto=TRUE`).
- **gamma.initAuto** A vector containing the initial values for parameters $\gamma$ when “reasonable” values are automatically computed (when `gamma.auto=TRUE`).
- **probs** An $N \times M \times K \times J$ array containing the estimated conditional probabilities $\hat{p}_{mkj} = \hat{p}_{mkj}(\gamma_{mj} + z'_{im} \alpha_m)$, where the first to fourth dimensions correspond to individuals, manifest variables, categories of manifest variables and latent classes, respectively.
prior An \( N \times J \) matrix containing the estimated latent class probabilities \( \hat{\pi}_{ij} = \hat{\pi}_j(x_i' \beta) \), where rows correspond to individuals and columns, to latent classes.

posterior An \( N \times J \) matrix containing the estimated posterior latent class probabilities \( h_{ij}(\hat{\phi}) \), where rows correspond to individuals and columns to latent classes.

predclass A vector of length \( N \) of predicted class memberships, by modal assignment.

P The respective size of each latent class, equal to the mean of the priors.

numiter The number of iterations required by the estimation algorithm to achieve convergence.

coeffBeta An \( P \times J \) matrix of estimated \( \beta_{pj} \), where rows correspond to covariates and columns, to latent classes.

param.se A vector containing the standard error of each estimated parameter, in the following order: \( \beta_{jp}, \gamma_{mjk}, \alpha_{mlk} \) where the last index always moves faster.

param.V The covariance matrix of the coefficient estimates (in the same order as in param.se).

coeffGamma An \( M \times (K-1) \) matrix of estimated parameters \( \gamma_{mjk} \). Each row corresponds to manifest variable \( m \). Within each row, columns correspond to latent classes \( j \) and categories of manifest variables \( k \) (except the last category, for which \( \gamma_{mjkM} = 0 \), the index of the latter moving faster.

coeffAlpha An \( M \times (K-1) \) matrix of estimated parameters \( \alpha_{mlk} \). Each row corresponds to manifest variable \( m \). Within each row, columns correspond to covariates \( l \) and categories of manifest variables \( k \) (except the last category, for which \( \alpha_{mlkM} = 0 \), the index of the latter moving faster.

meanProbs An \( M \times K \times J \) array of estimated conditional probabilities evaluated at the sample mean of the covariates. The first to third dimensions correspond to manifest variables, categories of manifest variables and latent classes, respectively.

eflag Logical, error flag. TRUE if estimation algorithm needed to automatically restart with new initial parameters, otherwise FALSE. A restart is caused in the event of computational/rounding errors that result in nonsensical parameter estimates.

npar The number of estimated parameters.

aic Value of the AIC criterion for the estimated model.

bic Value of the BIC criterion for the estimated model.

Nobs Number of fully observed cases.

x A dataframe containing the covariates for the latent class probabilities.

z A dataframe containing the covariates for the conditional probabilities.

y A dataframe containing the manifest variables.

identifiability A list containing the eigenvalues and the inverse condition number of the matrices involved in conditions (iii’) and (iv’) of Theorem 1 (Local Identifiability) in Huang and Bandeen-Roche (2004).

maxiter The maximum number of iterations of the estimation algorithm.

resid.df The number of residual degrees of freedom, equal to the lesser of \( N \) and \( MK \), minus npar.

time Computation time of model estimation.
**Note**

This function is an extension of the source code of the R package poLCA (Linzer and Lewis, 2011) to the methodology proposed by Huang and Bandeen-Roche (2004).

**References**


**Examples**

```r
## 2 models for a subset of dataset election in package poLCA
library("poLCA")
data("election", package = "poLCA")
election$GENDER <- factor(election$GENDER)
elec <- election[, c(1:3, 7:12, 16:17)]
elec <- na.omit(elec)
elec <- elec[1:200,]

## Model 1: 3 classes, 1 covariate for modelling latent class membership
fm1 <- cbind(MORAL, CAREG, KNOWG, MORALB, CARESB, KNOWB) ~ PARTY
polCA1 <- polCA(formula = fm1, data = elec, nclass = 3, nrep = 10)

## Model 2: 3 classes, 1 covariate in the model for latent class membership,
## 1 covariate in the model for the manifest variables probabilities
fm2 <- cbind(MORAL, CAREG, KNOWG, MORALB, CARESB, KNOWB) ~ PARTY
fm3 <- cbind(MORAL, CAREG, KNOWG, MORALB, CARESB, KNOWB) ~ 1 + GENDER
covLCA1 <- covLCA(formula1 = fm2, formula2 = fm3, data = elec, nclass = 3,
beta.auto = TRUE, gamma.auto = TRUE, alpha.auto = TRUE, maxit = 10000)

## Not run: 2 models for dataset election in package poLCA
library("poLCA")
data("election", package = "poLCA")
election$GENDER <- factor(election$GENDER)
elec <- election[, c(1:12, 16:17)]
elec <- na.omit(elec)

## Model 1: 3 classes, 1 covariate for modelling latent class membership
```
```r
fm1 <- cbind(MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG, MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB) ~ PARTY
polCA1 <- poLCA(formula=fm1, data=elec, nclass=3, nrep=10)

## Model 2: 3 classes, 1 covariate in the model for latent class membership,
## 1 covariate in the model for the manifest variables probabilities
fm2 <- cbind(MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG, MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB) ~ 1 + PARTY
fm3 <- cbind(MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG, MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB) ~ 1 + GENDER

covLCA1 <- covLCA(formula1=fm2, formula2=fm3, data=elec, nclass=3,
beta.auto=TRUE, gamma.auto=TRUE, alpha.auto=TRUE, maxit=10000)

## End(Not run)
```
Index

*Topic methods
  covLCA, 1

covLCA, 1