

# covLCA: An R function for the estimation of latent class models with covariate effects on underlying and measured variables

Aur lie Bertrand\*

Christian M. Hafner†

June 20, 2012

In the following we describe the arguments and outputs of the new R function `covLCA`. It is an extension of the source code of the R package `poLCA` (Linzer and Lewis, 2011a) to the methodology proposed by Huang and Bandeen-Roche (2004). Some elements of this document are adopted from the reference manual of package `poLCA` (Linzer and Lewis, 2011a) and from Linzer and Lewis (2011b).

## 1 Input

The input arguments are as follows:

```
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)
```

- **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**. 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 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.

---

\*ISBA, Universit  catholique de Louvain

†Corresponding author. ISBA and CORE, Universit  catholique de Louvain, Voie du Roman Pays 20, 1348 Louvain-la-Neuve, Belgium; [christian.hafner@uclouvain.be](mailto:christian.hafner@uclouvain.be)

- **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-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 indice 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 variables  $k$  (except the last category, for which  $\alpha_{mlK_m} = 0$ ), the indice 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  $\gamma_{mjK_m} = 0$ ), the indice of the latter moving faster. The default is **NULL**, leading either to an automatic search for “reasonable” initial values (when **gamma.auto=TRUE**, the default) or to the generation of random starting values (when **gamma.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  $(Y_{i1}, \mathbf{z}_{i1}), \dots, (Y_{iM}, \mathbf{z}_{iM})$  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  $(Y_{i1}, \mathbf{z}_{i1}), \dots, (Y_{iM}, \mathbf{z}_{iM})$  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 wheter **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  $\beta_{jp}$ ,  $\alpha_{mlk}$  and  $\gamma_{mjk}$ .

## 2 Output

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

- **llik**: the maximum value of the log-likelihood 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  $\beta$  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  $\hat{p}_{imkj} = \hat{p}_{mkj}(\gamma_{mj} + \mathbf{z}'_{im}\alpha_m)$ , 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  $\hat{\pi}_{ij} = \hat{\pi}_j(\mathbf{x}'_i\beta)$ , 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 indice always moves faster.
- **param.V**: the covariance matrix of the coefficient estimates (in the same order as in **param.se**).
- **coeffGamma**: an  $M \times J(K - 1)$  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  $\gamma_{mjK_m} = 0$ ), the indice of the latter moving faster.
- **coeffAlpha**: an  $M \times L(K - 1)$  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  $\alpha_{mlK_m} = 0$ ), the indice 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.
- `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.

### 3 Illustrative example

We estimate a latent class model with covariate effects on underlying and measured variables, using the sample dataset `election` which can be found in the R package `poLCA` (Linzer and Lewis, 2011a) and was analyzed by Linzer and Lewis (2011b). They estimated a latent class model assuming 3 classes, with an effect of the party identification (treated as a continuous variable) on the latent class membership. We now assume that the gender of the respondent could influence the rating he or she assigns to each item.

We first estimate the same model as that estimated in Linzer and Lewis (2011b), but only on individuals with no missing values in the variables of interest:

```
> library(poLCA)
> data(election)
> election$GENDER=factor(election$GENDER)
> elec=election[,c(1:12,16:17)]
> elec=na.omit(elec)
> 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)
```

We obtain the following output:

```
$MORALG
      Pr(1) Pr(2) Pr(3) Pr(4)
class 1: 0.6221 0.3350 0.0172 0.0258
class 2: 0.1373 0.6682 0.1802 0.0143
class 3: 0.1081 0.3832 0.3038 0.2048

$CARESG
      Pr(1) Pr(2) Pr(3) Pr(4)
```

class 1:	0.4858	0.4164	0.0534	0.0444
class 2:	0.0388	0.6138	0.2886	0.0589
class 3:	0.0356	0.2281	0.4501	0.2861

#### \$KNOWG

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.7189	0.2451	0.0040	0.032
class 2:	0.0712	0.8173	0.1025	0.009
class 3:	0.1436	0.5327	0.2556	0.068

#### \$LEADG

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.4720	0.4326	0.0643	0.0311
class 2:	0.0256	0.6280	0.3144	0.0320
class 3:	0.0278	0.1899	0.5137	0.2685

#### \$DISHONG

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.0518	0.0538	0.2890	0.6054
class 2:	0.0210	0.1412	0.5341	0.3037
class 3:	0.1876	0.3435	0.3078	0.1611

#### \$INTELG

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.7381	0.2219	0.0089	0.0311
class 2:	0.0698	0.8159	0.1003	0.0140
class 3:	0.1704	0.5597	0.2000	0.0698

#### \$MORALB

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.1610	0.3749	0.3163	0.1478
class 2:	0.0310	0.6326	0.3003	0.0361
class 3:	0.4477	0.5135	0.0313	0.0075

#### \$CARESB

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.0458	0.1490	0.3780	0.4272
class 2:	0.0047	0.3274	0.5083	0.1597
class 3:	0.2516	0.6185	0.1097	0.0202

#### \$KNOWB

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.1300	0.3503	0.2989	0.2209
class 2:	0.0121	0.6511	0.2920	0.0447
class 3:	0.3427	0.5913	0.0660	0.0000

\$LEADB

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.0743	0.3035	0.3882	0.2340
class 2:	0.0301	0.5884	0.3294	0.0521
class 3:	0.3850	0.5803	0.0287	0.0060

\$DISHONB

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.0914	0.3117	0.3517	0.2451
class 2:	0.0264	0.1855	0.5931	0.1950
class 3:	0.0163	0.0680	0.2923	0.6234

\$INTELB

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
class 1:	0.1877	0.3637	0.2698	0.1787
class 2:	0.0350	0.6646	0.2616	0.0388
class 3:	0.3765	0.5878	0.0357	0.0000

Estimated class population shares

0.2736 0.3859 0.3405

Predicted class memberships (by modal posterior prob.)

0.2769 0.3815 0.3415

=====  
Fit for 3 latent classes:  
=====

2 / 1

	Coefficient	Std. error	t value	Pr(> t )
(Intercept)	-1.16155	0.17989	-6.457	0
PARTY	0.57436	0.06401	8.973	0

3 / 1

	Coefficient	Std. error	t value	Pr(> t )
(Intercept)	-4.97967	0.32771	-15.195	0
PARTY	1.36762	0.08081	16.924	0

=====  
number of observations: 1300

number of estimated parameters: 112

residual degrees of freedom: 1188

maximum log-likelihood: -16222.32

AIC(3): 32668.65

BIC(3): 33247.7

X<sup>2</sup>(3): 34565227664 (Chi-square goodness of fit)

ALERT: estimation algorithm automatically restarted with new initial values

We then use the function covLCA to estimate the full model:

```
> 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)
```

The different components of the output can be called one by one:

```
> covLCA1$l1lik # log-likelihood  
[1] -16160.62  
> covLCA1$aic # AIC criterion  
[1] 32617.24  
> covLCA1$bic # BIC criterion  
[1] 33382.41  
  
> covLCA1$numiter # number of iterations  
[1] 156  
  
> covLCA1$time # time needed to estimate the model  
Time difference of 6.203863 mins  
  
> covLCA1$P # latent class probabilities  
Latent class 1 Latent class 2 Latent class 3  
0.3384395 0.3813964 0.2801641  
  
> round(covLCA1$meanProbs,4) # conditional probabilities  
, , Latent class 1  
  
Pr(1) Pr(2) Pr(3) Pr(4)  
MORALG 0.1073 0.3864 0.3003 0.2061  
CARESG 0.0346 0.2293 0.4512 0.2848  
KNOWG 0.1464 0.5282 0.2567 0.0687  
LEADG 0.0274 0.1931 0.5157 0.2638  
DISHONG 0.1885 0.3410 0.3075 0.1630  
INTELG 0.1729 0.5550 0.2012 0.0709  
MORALB 0.4512 0.5102 0.0317 0.0069  
CARESB 0.2540 0.6176 0.1086 0.0198  
KNOWB 0.3474 0.5892 0.0634 0.0000  
LEADB 0.3901 0.5763 0.0283 0.0054  
DISHONB 0.0148 0.0681 0.2921 0.6250  
INTELB 0.3811 0.5844 0.0345 0.0000
```



, , Latent class 2

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
MORALG	0.1348	0.6718	0.1797	0.0137
CARESG	0.0371	0.6149	0.2913	0.0567
KNOWG	0.0698	0.8199	0.1019	0.0083
LEADG	0.0257	0.6265	0.3179	0.0299
DISHONG	0.0205	0.1418	0.5365	0.3013
INTELG	0.0679	0.8212	0.0985	0.0124
MORALB	0.0301	0.6408	0.2982	0.0309
CARESB	0.0041	0.3355	0.5100	0.1504
KNOWB	0.0111	0.6622	0.2899	0.0368
LEADB	0.0297	0.5980	0.3277	0.0446
DISHONB	0.0226	0.1808	0.5988	0.1978
INTELB	0.0336	0.6754	0.2574	0.0336

, , Latent class 3

	Pr(1)	Pr(2)	Pr(3)	Pr(4)
MORALG	0.6089	0.3452	0.0199	0.0260
CARESG	0.4685	0.4273	0.0586	0.0456
KNOWG	0.7019	0.2580	0.0076	0.0324
LEADG	0.4547	0.4415	0.0724	0.0314
DISHONG	0.0514	0.0555	0.2912	0.6019
INTELG	0.7236	0.2346	0.0095	0.0324
MORALB	0.1584	0.3725	0.3163	0.1528
CARESB	0.0408	0.1463	0.3782	0.4348
KNOWB	0.1245	0.3448	0.3015	0.2291
LEADB	0.0704	0.2997	0.3887	0.2413
DISHONB	0.0908	0.3168	0.3482	0.2443
INTELB	0.1762	0.3630	0.2762	0.1845

The AIC and BIC criteria, as well as the value of the log-likelihood, can be used to compare both models. The estimated latent class and conditional probabilities (the latter being evaluated at the sample mean of the covariate Gender) are close to those obtained in the simple model without the effect of Gender. The same holds for the coefficients in the model relating the latent class probabilities to the party identification, as well as for the significance (which can be asserted using the estimated standard errors):

```
> covLCA1$coeffBeta
      1 vs 3      2 vs 3
(Intercept) -4.970079 -1.2058510
PARTY       1.354008  0.5732146

> round(covLCA1$param.se[1:4],4) # Standard error for the betas, column by column
```

```
[1] 0.3404 0.0809 0.1761 0.0613
```

```
> round(covLCA1$coeffAlpha,4)
      Var. 1, k=1 Var. 1, k=2 Var. 1, k=3
MORALG      0.6560      0.2806     -0.1350
CARESG      1.0578      0.4991      0.2018
KNOWG       0.0984     -0.1668      0.0906
LEADG       1.0417      0.5540      0.4652
DISHONG     -0.2012     -0.2625      0.1487
INTELG       0.0421     -0.1039      0.4068
MORALB       0.3666      0.2722      0.2915
CARESB       0.7487      0.1000      0.0259
KNOWB       0.4755      0.2444     -0.1219
LEADB       0.5215      0.2061      0.0807
DISHONB     -0.7163     -0.1224      0.1748
INTELB       0.7785      0.2514     -0.0510
```

```
> round(covLCA1$param.se[113:148],4) # Standard error for the alphas, row by row
[1] 0.2581 0.2283 0.2396 0.2615 0.1973 0.1893 0.3296 0.3135 0.3390 0.2838
[11] 0.2190 0.2006 0.2307 0.1802 0.1397 0.3193 0.3036 0.3388 0.2881 0.2595
[21] 0.2685 0.2752 0.1955 0.1713 0.2797 0.2387 0.2433 0.2808 0.2313 0.2282
[31] 0.3082 0.1769 0.1403 0.2832 0.2532 0.2626
```

The information (estimates and estimated standard errors) regarding the coefficients measuring the effect of the Gender on the conditional probabilities can be manually summarized in a table. It allows us to identify which parameters are significant (on the basis of the Normal distribution).

```
> round(cbind(covLCA1$coeffAlpha[,1],covLCA1$param.se[seq(from=113,by=3,length=12)] ,
+ covLCA1$coeffAlpha[,2],covLCA1$param.se[seq(from=114,by=3,length=12)] ,
+ covLCA1$coeffAlpha[,3],covLCA1$param.se[seq(from=115,by=3,length=12)]),4)
      [,1] [,2] [,3] [,4] [,5] [,6]
MORALG  0.6560 0.2581 0.2806 0.2283 -0.1350 0.2396
CARESG  1.0578 0.2615 0.4991 0.1973 0.2018 0.1893
KNOWG   0.0984 0.3296 -0.1668 0.3135 0.0906 0.3390
LEADG   1.0417 0.2838 0.5540 0.2190 0.4652 0.2006
DISHONG -0.2012 0.2307 -0.2625 0.1802 0.1487 0.1397
INTELG  0.0421 0.3193 -0.1039 0.3036 0.4068 0.3388
MORALB  0.3666 0.2881 0.2722 0.2595 0.2915 0.2685
CARESB  0.7487 0.2752 0.1000 0.1955 0.0259 0.1713
KNOWB   0.4755 0.2797 0.2444 0.2387 -0.1219 0.2433
LEADB   0.5215 0.2808 0.2061 0.2313 0.0807 0.2282
DISHONB -0.7163 0.3082 -0.1224 0.1769 0.1748 0.1403
INTELB  0.7785 0.2832 0.2514 0.2532 -0.0510 0.2626
```

The check on the four sufficient conditions for local identifiability given in Huang and Bandeen-Roche (2004) is implemented in the function `covLCA`: a warning message automatically appears at the end of the estimation process if at least one of the conditions does not

hold. There is a piece of output regarding the third and fourth conditions in Theorem 1 in Huang and Bandeen-Roche (2004): some vectors  $\boldsymbol{\tau}$  must be linearly independent (`tau.eigen` and `tau.invCond` are respectively the eigenvalues and inverse condition number of the matrix containing these vectors) and the design matrices containing the covariates  $\mathbf{x}$  and  $\mathbf{z}$  must have full column rank (`x.eigen` and `z.eigen` are their eigenvalues, while `x.invCond` and `z.invCond` are their inverse condition numbers).

```
> covLCA1$identifiability
$tau.eigen
[1] 1.911541e-04 2.357618e-05 6.637461e-06

$x.eigen
[1] 25714.4189 295.5811

$z.eigen
[1] 1780.0162 234.9838

$tau.invCond
[1] 0.1821579

$x.invCond
[1] 0.07436295

$z.invCond
[1] 0.3064111
```

## 4 Possible developments

This is a first version of this function: it has not been tested extensively in many different situations and it could be improved by different means:

- Writing some steps of the code in C (as is done in `poLCA`) would help speed up the estimation process.
- Letting  $K_m$ , the number of categories of each manifest variable, vary with the manifest variable  $m$  would make the model more flexible.
- Implementing another treatment of the missing values (for each individual, the computation of the likelihood based on the observed information) would allow us to keep all the available information when estimating the model.
- Creating an output with a clear presentation of all important results (estimates with their standard errors and their p-value in a table for example) would make the function more user-friendly.

## References

- [1] HUANG G.-H., BANDEEN-ROCHE K. (2004) Building an identifiable latent class model with covariate effects on underlying and measured variables. *Psychometrika*, Vol. 69, No. 1, 5-32.
- [2] LINZER D. A., LEWIS J. (2011a) poLCA: Polytomous Variable Latent Class Analysis. R package version 1.3.1. <http://userwww.service.emory.edu/~dlinzer/poLCA>.
- [3] LINZER D.A., LEWIS J. (2011b) poLCA: an R Package for Polytomous Variable Latent Class Analysis. *Journal of Statistical Software*, Vol. 42, No. 10, 1-29.