Package: DLCA (via r-universe)

October 20, 2026

Version 1.1

Date 2026-08-20

Title Divisive Latent Class Analysis

Author Daniel W. van der Palm [aut, cre], L. Andries van der Ark [ctb]

Maintainer Daniel W. van der Palm <danielvdpalm@gmail.com>

Imports Rcpp (>= 0.11.4)

LinkingTo Rcpp

SystemRequirements OpenMP

Description Provides algorithms for estimating divisive and standard latent class models. The divisive latent class method follows van der Palm, van der Ark and Vermunt (2016) <doi:10.1007/s00357-016-9195-5>. Both algorithms use expectation-maximization and Newton-Raphson optimization and are implemented in 'C++' for speed through 'Rcpp'.

License GPL (>= 2)

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2026-08-21 09:10:02 UTC

RemoteUrl https://github.com/cran/DLCA

RemoteRef HEAD

RemoteSha b1ffb77f49a1c238a78a3e1c4662e4ff191b94b7

Contents

checkDLCAInstall 2
DLC 2
DLCA-native 3
getUnique 4
LCA 5
prepareData 5
runDLCA 6

runLCA 7
runLCAPrepared 7
runSLCA 8
simData 9
simDataLCM 10

Index **12**

`checkDLCAInstall` *Inspect a DLCA Installation*

Description

Reports diagnostic information about the installed DLCA package and can optionally perform a small end-to-end fitting test.

Usage

```
checkDLCAInstall(runSmokeTest = FALSE)
```

Arguments

`runSmokeTest`	Logical. If `TRUE`, fit a small model to verify that the compiled code and high-level R interface work together.

Value

A list containing the package and shared-library paths, names masked in the global environment, an interface inspection result, and either `NULL` or the smoke-test result. The smoke-test result includes the random seed so the generated data can be reproduced.

`DLC` *The C++ code to estimate a divisive latent class model*

Description

The C++ code to estimate a divisive latent class model

Usage

```
DLC(xcpp, frequencies, ncats, settings, crtrm, verbose = FALSE)
```

Arguments

xcpp	The data matrix
frequencies	A vector with a frequency per unique response pattern
ncats	A vector with the number of categories per variable
settings	A list of settings pertaining to tier1 and tier2 starting sets
crtrm	The criterion value to decide whether to split or not
verbose	Logical. If TRUE, progress and diagnostic output is printed. Defaults to FALSE.

Value

A list of results

DLCA-native *Low-Level DLCA Native Interfaces*

Description

Direct interfaces to compiled DLCA routines. Most users should use runLCA, runLCAPrepared, or runSLCA, which validate and prepare inputs and format results.

Usage

```
CalculateFitStatistics(npar, logLikelihood, sampleSize, entropy, nclass)

IndependenceLogLikelihood(xcpp, frequencies, ncats)

LCAWarmStart(nclass, xcpp, frequencies, ncats, settings, cprobStart,
  condpStart, returnPosteriors = TRUE, verbose = FALSE)

RunLCA(verbose = FALSE)

RunSingleClassLCA(nclass, xcpp, verbose = FALSE)

SLCASequence(xcpp, frequencies, ncats, sequenceSettings,
  startSettingOverrides, criterion, warmStarts,
  returnPosteriors = TRUE, verbose = FALSE)
```

Arguments

npar	Number of estimated model parameters.
logLikelihood	Model log-likelihood.
sampleSize	Effective sample size.
entropy	Classification entropy.
nclass	Number of latent classes.

xcpp	Numeric response matrix in the internal zero-based representation; missing values are represented by 99.
frequencies	Positive frequency for each row of xcpp.
ncats	Number of response categories for each column of xcpp.
settings	Integer vector containing the tier-1 set count, tier-1 iteration count, tier-2 set count, and tier-2 iteration count.
cprobStart	Initial latent class proportions.
condpStart	Initial conditional response probabilities, represented as a matrix with nclass * ncol(xcpp) rows.
returnPosteriors	Logical. If TRUE, calculate and return posterior class probabilities.
verbose	Logical. If TRUE, print progress and diagnostics.
sequenceSettings	Integer vector containing stepSize, startNclass, and maxNclass.
startSettingOverrides	Four-element integer vector overriding start-set settings; use zero to select a default.
criterion	Character model-selection criterion.
warmStarts	Logical. If TRUE, use adjacent-model warm starts.

Value

The return value depends on the entry point. CalculateFitStatistics returns a named list of fit statistics, and IndependenceLogLikelihood returns a numeric log-likelihood. LCAWarmStart, RunSingleClassLCA, and SLCASequence return model-result lists. RunLCA is a legacy development entry point that returns a numeric log-likelihood.

See Also

LCA, runLCA, runSLCA

---

getUnique *The C++ code to obtain the unique response patterns*

Description

The C++ code to obtain the unique response patterns

Usage

```
getUnique(xcpp)
```

Arguments

xcpp	The data matrix

Value

A list with data matrix with unique response patterns and a frequency vector

---

`LCA` *The C++ code to estimate a latent class model*

Description

The C++ code to estimate a latent class model

Usage

```
LCA(nclass, xcpp, frequencies, ncats, settings, returnPosteriors = TRUE,
  verbose = FALSE)
```

Arguments

`nclass`	The specified number of latent classes to be estimated
`xcpp`	The data matrix
`frequencies`	A vector with a frequency per unique response pattern
`ncats`	A vector with the number of categories per variable
`settings`	A list of settings pertaining to tier1 and tier2 starting sets
`returnPosteriors`	Logical. If TRUE, posterior class probabilities are returned.
`verbose`	Logical. If TRUE, progress and diagnostic output is printed. Defaults to FALSE.

Value

A list of results

`prepareData` *Prepares the data to avoid estimation issues and to increase computation speed*

Description

prepareData only retains the unique response patterns, and it counts the number of times each pattern occurs. In this way, the data structure that is passed to the C++ code typically is smaller than the original. Therefore, the computation time can be reduced. Furthermore, in the C++ EM algorithm, data-values are used to directly access a parameter array, to prevent look-ups. In order for this to work, there cannot be any gaps in the item values. Therefore, as an example, [0,1,3] is changed to [0,1,2]. Also, if the item values are ["horse","cow","chicken"] then these are changed to [0,1,2] as well for convenience during the estimation procedure.

Usage

```
prepareData(datmat, verbose = FALSE)
```

Arguments

datmat	The data matrix
verbose	Logical. If TRUE, diagnostic messages are printed. Defaults to FALSE.

Value

The prepared data, frequency column and number of categories per variable

---

runDLCA *Estimate a divisive latent class model*

Description

Estimate a divisive latent class model

Usage

```
runDLCA(dataMatrix, tier1StartSets = 50, tier1StartIterations = 250,
  tier2StartSets = 10, tier2StartIterations = 500, criterion = "DLL",
  minDLL = 1, verbose = FALSE)
```

Arguments

dataMatrix	The matrix containing the data
tier1StartSets	The number of random starting sets used at every division in the first round (tier1)
tier1StartIterations	The number of EM iterations used for each starting set in the first round (tier1)
tier2StartSets	The number of random starting sets used at every division in the second round (tier2)
tier2StartIterations	The number of EM iterations used for each starting set in the second round (tier2)
criterion	The criterion used to decide whether to divide or not, typically a fit-criterion
minDLL	The minimum increase in the log-likelihood that is used if criterion is set to "DLL"
verbose	Logical. If TRUE, progress and diagnostic output is printed. Defaults to FALSE.

Value

A list of results with the number of classes, latent class proportions, conditional response array, posterior class probabilities and the model log-likelihood

runLCA *Estimate a latent class model*

Description

Estimate a latent class model

Usage

```
runLCA(dataMatrix, nclass, tier1StartSets = NULL,
  tier1StartIterations = NULL, tier2StartSets = NULL,
  tier2StartIterations = NULL, returnPosteriors = TRUE, verbose = FALSE)
```

Arguments

dataMatrix	The matrix containing the data
nclass	The specified number of latent classes to be estimated
tier1StartSets	The number of random starting sets used at every division in the first round (tier1)
tier1StartIterations	The number of EM iterations used for each starting set in the first round (tier1)
tier2StartSets	The number of random starting sets used at every division in the second round (tier2)
tier2StartIterations	The number of EM iterations used for each starting set in the second round (tier2)
returnPosteriors	Logical. If TRUE, posterior class probabilities are returned.
verbose	Logical. If TRUE, progress and diagnostic output is printed. Defaults to FALSE.

Value

A list of results with latent class proportions, conditional response array, posterior class probabilities and the model log-likelihood

runLCAPrepared *Estimate a Latent Class Model from Prepared Data*

Description

Fits a latent class model using data that have already been validated and compressed by prepareData. This avoids repeating data preparation when fitting several models to the same data.

Usage

```
runLCAPrepared(dataCollection, nclass, tier1StartSets = NULL,
  tier1StartIterations = NULL, tier2StartSets = NULL,
  tier2StartIterations = NULL, returnPosteriors = TRUE, verbose = FALSE)
```

Arguments

`dataCollection`	A validated data collection returned by `prepareData`.
`nclass`	The number of latent classes to estimate.
`tier1StartSets`	Optional number of starting sets used in tier 1.
`tier1StartIterations`	Optional number of EM iterations per tier-1 starting set.
`tier2StartSets`	Optional number of retained starting sets used in tier 2.
`tier2StartIterations`	Optional number of EM iterations per tier-2 starting set.
`returnPosteriors`	Logical. If `TRUE`, return posterior class probabilities.
`verbose`	Logical. If `TRUE`, print progress and diagnostics.

Value

A list containing class proportions, conditional response probabilities, posterior probabilities when requested, the log-likelihood, and fit statistics.

See Also

`runLCA`, `prepareData`

---

runSLCA *Estimate a Sequence of Latent Class Models*

Description

Fits latent class models with increasing numbers of classes and selects a model using the requested fit criterion.

Usage

```
runSLCA(dataMatrix, stepSize = 1, startNclass = 1, maxNclass = 100,
  criterion = "AIC", tier1StartSets = NULL,
  tier1StartIterations = NULL, tier2StartSets = NULL,
  tier2StartIterations = NULL, warmStarts = TRUE,
  returnPosteriors = TRUE, verbose = FALSE)
```

Arguments

dataMatrix	A matrix or data frame containing categorical responses.
stepSize	The increase in the number of latent classes between successive models.
startNclass	The number of classes in the first model.
maxNclass	The maximum number of classes. Use zero for no explicit maximum.
criterion	The model-selection criterion. Supported values include "AIC", "AIC3", "BIC", "CAIC", "SABIC", their entropy-adjusted variants, "ICLBIC", "ICL-BIC", "entropy", and "normalizedEntropy".
tier1StartSets	Optional number of starting sets used in tier 1.
tier1StartIterations	Optional number of EM iterations per tier-1 starting set.
tier2StartSets	Optional number of retained starting sets used in tier 2.
tier2StartIterations	Optional number of EM iterations per tier-2 starting set.
warmStarts	Logical. If TRUE, adjacent models reuse a split of the preceding solution as a warm-start candidate.
returnPosteriors	Logical. If TRUE, return posterior class probabilities for the selected model.
verbose	Logical. If TRUE, print progress and diagnostics.

Value

A list describing the selected model, including its number of classes, parameters, posterior probabilities when requested, fit statistics, and a fitSummary data frame for the fitted sequence.

See Also

runLCA, runLCAPrepared

---

simData *Simulate data to try out the runDLCA or runLCA function*

Description

Simulate data to try out the runDLCA or runLCA function

Usage

```
simData(N, J, perc.miss)
```

Arguments

N	The sample size (rows)
J	The number of variables (columns)
perc.miss	The proportion of missingness (.001 - .999)

Value

A data matrix with N N rows and J columns

Examples

```
simData(1000, 8, .05)
simData(500, 12, .20)
```

---

simDataLCM *Simulate Data from a Latent Class Model*

Description

simDataLCM randomly generates model parameters and observations. simDataLCMFromParameters generates observations from parameters supplied by the user.

Usage

```
simDataLCM(N, J, K, M, classAlpha = 1, responseAlpha = 1, seed = NULL)

simDataLCMFromParameters(N, classProportions, condResponseProbs,
  ncat = NULL, seed = NULL)
```

Arguments

N	The number of observations.
J	The number of observed variables.
K	The number of latent classes.
M	The number of response categories per variable. Supply one integer or an integer vector of length J.
classAlpha	A positive Dirichlet concentration parameter, supplied as one value or one value per class.
responseAlpha	A positive Dirichlet concentration parameter used for conditional response probabilities.
seed	An optional seed passed to set.seed. If NULL, the current R random-number state is used.
classProportions	A numeric vector of latent class probabilities that sums to one.
condResponseProbs	A numeric array indexed by variable, response category, and latent class.
ncat	Optional integer vector containing the active number of categories for each variable. It is inferred when omitted.

Value

A list containing the simulated data, sampled class memberships, class proportions, conditional response probabilities, and category counts.

See Also

simData, runLCA

Examples

```
sim <- simDataLCM(N = 100, J = 4, K = 2, M = 3, seed = 1)
head(sim$data)

cprob <- c(0.4, 0.6)
condp <- array(NA_real_, dim = c(2, 2, 2))
condp[, , 1] <- matrix(c(0.8, 0.2,
                         0.3, 0.7), nrow = 2, byrow = TRUE)
condp[, , 2] <- matrix(c(0.4, 0.6,
                         0.7, 0.3), nrow = 2, byrow = TRUE)
sim2 <- simDataLCMFromParameters(50, cprob, condp, seed = 2)
```

Index

CalculateFitStatistics *(*DLCA-native*)*, 3
checkDLCAInstall, 2

DLC, 2
DLCA-native, 3

getUnique, 4

IndependenceLogLikelihood *(*DLCA-native*)*, 3

LCA, *4*, 5
LCAWarmStart *(*DLCA-native*)*, 3

prepareData, 5, *7, 8*

runDLCA, 6
RunLCA *(*DLCA-native*)*, 3
runLCA, *3, 4*, 7, *8, 9*, *11*
runLCAPrepared, *3*, 7, *9*
RunSingleClassLCA *(*DLCA-native*)*, 3
runSLCA, *3, 4*, 8

set.seed, *10*
simData, 9, *11*
simDataLCM, 10
simDataLCMFromParameters *(*simDataLCM*)*, 10
SLCASequence *(*DLCA-native*)*, 3