Package 'tseLCA'

July 11, 2026

Type Package

Title Three-Step Estimation for Latent Class Analysis

Version 1.0.2

Description Implements BCH (Bolck-Croon-Hagenaars) <doi:10.1093/pan/mph001> and ML (Vermunt's maximum likelihood) <doi:10.1093/pan/mpq025> approaches for three-step estimation of latent class models with covariates and distal outcomes, following Bakk, Tekle & Vermunt (2013) <doi:10.1177/0081175012470644>, Bakk, Oberski & Vermunt (2014) <https://www.jstor.org/stable/24573086>, and Bakk & Kuha (2018) <doi:10.1007/s11336-017-9592-7>. Built on 'multilevLCA' (Lyrvall et al., 2025) <doi:10.1080/00273171.2025.2473935> for Step-1 measurement model estimation, this package extends it with support for Gaussian, Poisson, and binomial distal outcome families. Unlike 'poLCA', which relies on one-step estimation and cannot accommodate a measurement model from a different sample, this package uses a stepwise approach to prevent the structural model from influencing latent class formation. Implements correct sandwich variance estimation that propagates measurement uncertainty from the first-step through classification-error correction in the final step (Bakk, Oberski & Vermunt, 2014). Supports polytomous items and missing data in the measurement model with full information maximum likelihood. A data-generating process replicating the Bakk & Kuha (2018) simulation study is included.

License GPL (>= 3)

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports multilevLCA, cli

Suggests poLCA, testthat (>= 3.0.0), parallel, knitr, rmarkdown, spelling

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

URL https://samleebyu.github.io/tseLCA/, https://github.com/SamLeeBYU/tseLCA

BugReports https://github.com/SamLeeBYU/tseLCA/issues

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Sam Lee [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0007-2977-6661>),
Jay Goodliffe [ctb]

Maintainer Sam Lee <samlee@arizona.edu>

Repository CRAN

Date/Publication 2026-07-11 08:40:02 UTC

Contents

`bk2018_params` *Default population parameters for the Bakk & Kuha (2018) simulation*

Description

A list of pre-specified parameters used by `generate_data()` and related functions. All elements correspond to the values stated in the paper (Section 3, p. 879).

Usage

```
bk2018_params
```

Details

`class_props` Length-3 vector of equal class proportions (1/3 each).

`separation_levels` Named vector mapping `"low"`, `"mid"`, `"high"` to the probability of a "likely" response (0.70, 0.80, 0.90).

`covariate_params` List with `$b0` (intercepts) and `$b` (slopes) for the multinomial logit P(X=t | Zp). Intercepts `b02` and `b03` are set so that marginal class sizes average to 1/3 when Zp ~ Uniform{1..5}.

`distal_params` List with `$mu` (class means, c(-1, 1, 0)) and `$sigma` (residual SD = 1) for the distal outcome model.

Value

A named list with four elements:

`class_props` Length-3 numeric vector of equal class proportions (1/3 each).

`separation_levels` Named numeric vector mapping `"low"`, `"mid"`, `"high"` to 0.70, 0.80, 0.90.

`covariate_params` List with `$b0` (intercepts) and `$b` (slopes) for the multinomial logit P(X=t|Zp).

`distal_params` List with `$mu` (class means) and `$sigma` (residual SD).

Examples

```
# True item-response probabilities for high separation
bk2018_params$rho_high

# Covariate model parameters (intercepts and slopes)
bk2018_params$covariate_params

# Distal outcome parameters
bk2018_params$distal_params
```

`coef.tseLCA_measurement`

Extract coefficients from a tseLCA model object

Description

Extract coefficients from a tseLCA model object

Usage

```
## S3 method for class 'tseLCA_measurement'
coef(object, ...)

## S3 method for class 'tseLCA_covariate'
coef(object, which = c("three_step", "two_step"), ...)
```

```
## S3 method for class 'tseLCA_distal'
coef(object, ...)

## S3 method for class 'tseLCA_both'
coef(
  object,
  which = c("both", "covariate", "distal"),
  step = c("three_step", "two_step"),
  ...
)
```

Arguments

object	A tseLCA object returned by three_step().
...	Further arguments (currently unused).
which	Character. For covariate and both models: "three_step" (default) or "two_step". For both models also accepts "covariate", "distal", or "both".
step	Character. For tseLCA_both: "three_step" (default) or "two_step".

Value

The coefficient matrix (covariate models), named numeric vector (distal models), or a named list of both (measurement or both models).

Examples

```
d    <- generate_data(100, "high", "covariate", seed = 1)
fit_m <- three_step(d, paste0("Y", 1:6), n_classes = 3)
coef(fit_m)   # returns list with $prevalences and $item_probs

d   <- generate_data(200, "high", "covariate", seed = 1)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", use.simple.cov = TRUE)
coef(fit)                      # three-step estimates
coef(fit, which = "two_step")  # two-step starting values


d   <- generate_data(200, "high", "distal", seed = 2)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zo.name = "Zo", use.simple.cov = TRUE)
coef(fit)   # named vector of class means


d   <- generate_data(200, "high", "covariate", seed = 1)
d$Zo <- rnorm(200, mean = c(-1, 0, 1)[d$X], sd = 1)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", Zo.name = "Zo",
                  use.simple.cov = TRUE)
coef(fit, which = "covariate")
coef(fit, which = "distal")
```

```
coef(fit, which = "both")
```

`draw_classes` *Draw latent class memberships from their marginal distribution*

Description

Draw latent class memberships from their marginal distribution

Usage

```
draw_classes(n, pi_)
```

Arguments

`n`	Integer. Sample size.
`pi_`	Numeric vector of length T. Class proportions (must sum to 1).

Value

Integer vector of length n with values in `1:T`.

Examples

```
# Draw 100 class labels from equal prevalences
draw_classes(100, c(1/3, 1/3, 1/3))
```

`draw_classes_given_Zp` *Draw latent classes conditional on the covariate (scenario "covariate")*

Description

Draw latent classes conditional on the covariate (scenario "covariate")

Usage

```
draw_classes_given_Zp(Zp, params)
```

Arguments

`Zp`	Numeric vector of length n. Covariate values.
`params`	Multinomial logistic parameter list (see `bk2018_params$covariate_params`).

Value

Integer vector of length n with class labels in `1:T`.

Examples

```
Zp <- draw_Zp(1000)
X  <- draw_classes_given_Zp(Zp, bk2018_params$covariate_params)
table(X) # Should be roughly uniform
```

`draw_indicators` *Draw binary indicators given true class memberships*

Description

Draw binary indicators given true class memberships

Usage

```
draw_indicators(X, rho)
```

Arguments

`X`	Integer vector of length n. True latent class (1-indexed).
`rho`	T x K matrix. `rho[t, k] = P(Y_k = 1 | X = t)`.

Value

An n x K integer matrix of 0/1 values.

Examples

```
rho <- make_rho(0.9)
X   <- draw_classes(50, c(1/3, 1/3, 1/3))
draw_indicators(X, rho)
```

draw_Zo — *Draw a continuous distal outcome given true class memberships (scenario "distal")*

Description

Draw a continuous distal outcome given true class memberships (scenario "distal")

Usage

```
draw_Zo(X, params)
```

Arguments

X	Integer vector of length n. Latent class (1-indexed).
params	List with $mu (length-T class means) and $sigma (SD). See bk2018_params$distal_params.

Value

Numeric vector of length n.

Examples

```
X  <- draw_classes(100, c(1/3, 1/3, 1/3))
Zo <- draw_Zo(X, bk2018_params$distal_params)
tapply(Zo, X, mean)   # should be close to true mu
```

draw_Zp — *Draw the covariate Zp ~ Uniform{1, 2, 3, 4, 5}*

Description

Draw the covariate Zp ~ Uniform{1, 2, 3, 4, 5}

Usage

```
draw_Zp(n)
```

Arguments

n	Integer. Sample size.

Value

Integer vector of length n.

Examples

```
Zp <- draw_Zp(100)
table(Zp)
```

`fitZ_from_fit0` *Estimate covariate effects with measurement parameters fixed (two-step EM)*

Description

Fixes `mPhi` at `fit0$mPhi` and estimates multinomial logit coefficients `mGamma` (Q x (T-1)) via an EM algorithm with a BFGS M-step.

Usage

```
fitZ_from_fit0(
  fit0,
  data,
  Y.names,
  Zp.names,
  tol = 1e-06,
  maxIter = 200L,
  incomplete = FALSE,
  include.intercept = TRUE,
  rebase = "C1",
  starting_val = NULL,
  verbose = FALSE
)
```

Arguments

`fit0`	Output of `lca_step1()$fit0`.
`data`	A data.frame.
`Y.names`	Character vector of item column names.
`Zp.names`	Character vector of covariate column names.
`tol`	Convergence tolerance. Default `1e-6`.
`maxIter`	Maximum EM iterations. Default `200`.
`incomplete`	Logical. FIML for partially missing indicators. See the `Missing Data` section of `vignette("tseLCA", package = "tseLCA")`. Default `FALSE`.
`include.intercept`	Logical. Prepend intercept to covariate design matrix. Default `TRUE`.
`rebase`	Character or integer. Reference class for the multinomial logit parameterization (e.g. `"C1"`, `"C2"`, or an integer). Default `"C1"`. Must match the `rebase` used in `lca_step1()` so class column ordering is consistent.
`starting_val`	Optional Q x (T-1) starting value matrix for `mGamma`.
`verbose`	Logical. Print convergence messages. Default `FALSE`.

Value

A list with the following elements:

`mGamma` Q x (T-1) numeric matrix of multinomial logit coefficients, where Q is the number of columns in the covariate design matrix (including intercept if `include.intercept = TRUE`). Rows are named by covariate, columns by non-reference class (e.g. `"C2"`, `"C3"`).

`mPhi` Expanded item parameter matrix (items x classes), fixed at `fit0$mPhi` throughout estimation.

`vOmega` Length-T vector of marginal class proportions implied by the final `mGamma`, computed as column means of the fitted class probability matrix.

`LLKSeries` Single-column matrix of observed-data log-likelihoods, one row per EM iteration. Useful for diagnosing convergence.

`converged` Logical. `TRUE` if the EM loop exited before `maxIter` iterations or if the final log-likelihood change was below `tol`.

`n_obs` Integer. Number of observations used in estimation after listwise deletion on covariates.

Examples

```
d  <- generate_data(200, "high", "covariate", seed = 1)
s1 <- lca_step1(d, Y.names = paste0("Y", 1:6), n_classes = 3)

# Estimate two-step gamma with mPhi fixed at Step-1 values
fZ <- fitZ_from_fit0(
  fit0     = s1$fit0,
  data     = d,
  Y.names  = paste0("Y", 1:6),
  Zp.names = "Zp",
  verbose  = TRUE
)
fZ$mGamma   # Q x (T-1) coefficient matrix
fZ$converged
```

---

fitZ_from_multiLCA *Estimate two-step covariate model via multilevLCA (optional reference path)*

Description

Calls `multilevLCA::multiLCA` with `fixedpars = 1` and `Z = Zp.names` to fit the two-step covariate model. This is the original multilevLCA approach and is used when `get.twostep.vcov = TRUE` in `three_step()` to obtain multilevLCA's corrected standard errors for the two-step gamma estimates.

Usage

```
fitZ_from_multiLCA(
  data,
  Y.names,
  n_classes,
  Zp.names,
  maxIter.measurement,
  measurement.tol,
  covariate.tol,
  iter.measurement,
  R2.threshold,
  incomplete = FALSE,
  rebase = "C1",
  verbose = FALSE
)
```

Arguments

`data`	A data.frame.
`Y.names`	Character vector of item column names.
`n_classes`	Integer. Number of latent classes.
`Zp.names`	Character vector of covariate column names.
`maxIter.measurement`	Maximum EM iterations.
`measurement.tol`	Convergence tolerance.
`covariate.tol`	NR tolerance for the covariate model.
`iter.measurement`	Number of random restarts.
`R2.threshold`	Entropy R^2 restart threshold.
`incomplete`	Logical. FIML for partially missing indicators. See the `Missing Data` section of `vignette("tseLCA", package = "tseLCA")`. Default `FALSE`.
`rebase`	Character or integer. Reference class for column naming of `$mGamma`. Must match the `rebase` used in `three_step()` so coefficient labels are consistent. Default `"C1"`.
`verbose`	Logical.

Value

A list with the following elements:

`mGamma` Q x (T-1) numeric matrix of multinomial logit coefficients. Rows are named by covariate (including `"Intercept"`), columns by non-reference class (e.g. `"C2"`, `"C3"`).

`mPhi` Item parameter matrix (items x classes) from the fixed-parameter multilevLCA fit.

`vOmega` Length-T vector of marginal class proportions, computed as the average of the fitted class probability matrix (`vPi_avg` in multilevLCA output).

`LLKSeries` Matrix of observed-data log-likelihoods across EM iterations, passed through directly from the multilevLCA fit.

`raw_fit` The full `multilevLCA::multiLCA()` output object, including `$Varmat_cor` (corrected variance matrix) and `$SEs_cor_gamma` (corrected standard errors for `mGamma`) if available.

Examples

```
d <- generate_data(200, "high", "covariate", seed = 1)

# Two-step estimation via multiLCA (fixedpars = 1)
fZ_ml <- fitZ_from_multiLCA(
  data                = d,
  Y.names             = paste0("Y", 1:6),
  n_classes           = 3,
  Zp.names            = "Zp",
  maxIter.measurement = 5000L,
  measurement.tol     = 1e-8,
  covariate.tol       = 1e-6,
  iter.measurement    = 10L,
  R2.threshold        = 0.70
)
fZ_ml$mGamma           # two-step estimates
fZ_ml$raw_fit$Varmat_cor   # multilevLCA corrected vcov
```

`generate_all_conditions` *Generate datasets for all 18 conditions in the simulation design*

Description

Iterates over the 2 scenarios x 3 separation levels x 3 sample sizes, generating `n_rep` independent replications per condition. Seeds are derived deterministically from `base_seed` so the entire experiment is reproducible from a single integer.

Usage

```
generate_all_conditions(
  n_rep = 500L,
  base_seed = 5262026L,
  params = bk2018_params,
  scenarios = c("covariate", "distal"),
  sep_levels = c("low", "mid", "high"),
  sample_sizes = c(500L, 1000L, 2000L),
  verbose = TRUE
)
```

Arguments

n_rep	Integer. Replications per condition (paper uses 500).
base_seed	Integer. Base seed for reproducibility.
params	Population parameters list. Defaults to bk2018_params.
scenarios	Character. Lists the scenario(s) ("covariate" and/or "distal") wanting to be simulated. Passed into generate_data().
sep_levels	Character. Lists the separation level(s) ("low", "mid", "high") wanting to be simulated. Passed into generate_data().
sample_sizes	Integer. Lists the sample size(s) wanting to be generated for each replication condition. Passed into generate_data().
verbose	Logical. If TRUE (default), display a live CLI progress bar with per-rep status and ETA.

Value

Nested list indexed as datasets[[scenario]][[separation]][[as.character(n)]], each element a list of n_rep data frames.

Examples

```
# Generate 5 replicates for mid and high separation only
datasets <- generate_all_conditions(n_rep = 5L, base_seed = 1L,
                                    sep_levels = c("mid", "high"))
# Access a single replicate
head(datasets[["covariate"]][["high"]][["500"]][[1]])
```

---

generate_data *Generate one dataset following the Bakk & Kuha (2018) simulation design*

Description

Generate one dataset following the Bakk & Kuha (2018) simulation design

Usage

```
generate_data(
  n,
  separation = c("low", "mid", "high"),
  scenario = c("covariate", "distal"),
  params = bk2018_params,
  seed = NULL
)
```

Arguments

n	Integer. Sample size (paper uses 500, 1000, or 2000).
separation	Character. One of "low", "mid", "high". Maps to pi = 0.70, 0.80, 0.90 respectively.
scenario	Character. One of: "covariate" Zp (discrete, 1-5) predicts latent X via multinomial logit. "distal" Latent X predicts continuous Zo via linear regression.
params	List of population parameters. Defaults to bk2018_params.
seed	Integer or NULL. Optional random seed for reproducibility.

Value

A data.frame with columns:

Y1 .. Y6 Binary indicators (always present).

X True latent class, integer 1-3 (not observed in practice).

Zp Integer covariate 1-5 (scenario "covariate" only).

Zo Continuous distal outcome (scenario "distal" only).

Examples

```
# Covariate scenario with high separation
d <- generate_data(n = 200, separation = "high", scenario = "covariate",
                   seed = 1)
head(d)
colMeans(d)

# Distal outcome scenario
d2 <- generate_data(n = 200, separation = "high", scenario = "distal",
                    seed = 2)
head(d2)
```

---

lca_step1 *Fit the LCA measurement model (Step 1)*

Description

Estimates the latent class measurement model with **multilevLCA** and optionally, fixes mPhi and estimates covariate effects (two-step initialization) with fitZ_from_fit0().

Usage

```
lca_step1(
  data,
  Y.names,
  n_classes,
  Zp.names = NULL,
  maxIter.measurement = 5000L,
  measurement.tol = 1e-08,
  covariate.tol = 1e-06,
  iter.measurement = 10L,
  R2.threshold = 0.7,
  use.two.step = TRUE,
  estimate.one.step = TRUE,
  incomplete = FALSE,
  maxIter.fitZ = 200L,
  include.intercept = TRUE,
  rebase = "C1",
  verbose = FALSE
)
```

Arguments

`data`	A data.frame containing at minimum the indicator columns.
`Y.names`	Character vector of item column names.
`n_classes`	Integer. Number of latent classes.
`Zp.names`	Character vector of covariate column names, or `NULL`.
`maxIter.measurement`	Maximum EM iterations before giving up on convergence. Default `5000L`.
`measurement.tol`	Convergence tolerance. Default `1e-8`.
`covariate.tol`	Convergence tolerance for the `fitZ` M-step. Default `1e-6`.
`iter.measurement`	Number of random restarts when entropy R^2 is low. Default `10`.
`R2.threshold`	Entropy R^2 below which restarts are triggered. Default `0.7`.
`use.two.step`	Logical. If `TRUE`, also estimate `fitZ` with `fitZ_from_fit0()` if `Zp.names` is applied. Default `TRUE`.
`estimate.one.step`	Logical. If `FALSE`, skip the unconditional EM and only compute `fitZ`. Default `TRUE`.
`incomplete`	Logical. FIML for partially missing indicators. See the `Missing Data` section of `vignette("tseLCA", package = "tseLCA")`. Default `FALSE`.
`maxIter.fitZ`	Maximum EM iterations for `fitZ_from_fit0()`. Default `200`.
`include.intercept`	Logical. Prepend intercept to covariate design matrix. Default `TRUE`.

rebase	Character or integer specifying the reference latent class. Use "C1", "C2", etc. or an integer index. Default "C1". The measurement model is permuted so this class becomes column 1, making it the reference for all downstream multinomial logit parameterizations.
verbose	Logical. Print progress messages. Default FALSE.

Value

A list with $fit0 (multilevLCA::multiLCA() measurement model) and $fitZ (two-step covariate model from fitZ_from_fit0(), or NULL).

Examples

```
d <- generate_data(200, "high", "covariate", seed = 1)

# Measurement model only
s1 <- lca_step1(d, Y.names = paste0("Y", 1:6), n_classes = 3)
s1$fit0$vPi    # estimated class prevalences
s1$fit0$mPhi   # item-response probabilities

# With two-step covariate initialization
s1z <- lca_step1(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                 Zp.names = "Zp", use.two.step = TRUE, verbose = TRUE)
s1z$fitZ$mGamma   # two-step gamma estimates
```

---

make_rho *Build the item-response probability matrix for the simulation*

Description

Returns a T x K matrix rho where rho[t, k] = P(Y_k = 1 | X = t).

Usage

```
make_rho(pi_)
```

Arguments

pi_	Numeric scalar in (0.5, 1). Probability of the "likely" response. Use bk2018_params$separation_level
	for the three simulation levels.

Details

The three-class structure is:

- Class 1: pi_ on all 6 items (high responders).
- Class 2: pi_ on items 1-3, 1 - pi_ on items 4-6 (mixed).
- Class 3: 1 - pi_ on all 6 items (low responders).

Value

A 3 x 6 numeric matrix.

Examples

```
# High separation: P(Y=1|class) = 0.9 for the "high" class
make_rho(0.9)

# Low separation
make_rho(0.7)
```

`mnl_probs` *Compute multinomial logistic class probabilities given covariates*

Description

Evaluates P(X = t | Zp) for each observation using a multinomial logit with one or more covariates and class-specific intercepts and slopes.

Usage

```
mnl_probs(Zp, params)
```

Arguments

`Zp`	Numeric vector of length n, or numeric matrix of dimension n x P, where P is the number of covariates. A vector is treated as a single covariate (P = 1).
`params`	List with elements `$b0` (length-T intercepts, reference = 0) and `$b` (length-T slopes when P = 1, or P x T slope matrix when P > 1, reference class = 1). See `bk2018_params$covariate_params`.

Value

An n x T matrix of class probabilities (rows sum to 1).

Examples

```
# Single covariate: class membership probabilities for Zp = 1..5
mnl_probs(1:5, bk2018_params$covariate_params)

# Multiple covariates (n = 5, P = 2)
Zp_mat <- matrix(rnorm(10), nrow = 5, ncol = 2)
params2 <- list(b0 = c(0, 0.5, -0.5), b = matrix(rnorm(6), nrow = 2, ncol = 3))
mnl_probs(Zp_mat, params2)
```

plot.tseLCA_measurement *Plot item-response probability profiles for a tseLCA model*

Description

Delegates to `plot.multiLCA` from **multilevLCA**, which draws the class-specific item-response probability profiles from the Step-1 measurement model.

Usage

```
## S3 method for class 'tseLCA_measurement'
plot(x, horiz = FALSE, clab = NULL, ...)

## S3 method for class 'tseLCA_covariate'
plot(x, horiz = FALSE, clab = NULL, ...)

## S3 method for class 'tseLCA_distal'
plot(x, horiz = FALSE, clab = NULL, ...)

## S3 method for class 'tseLCA_both'
plot(x, horiz = FALSE, clab = NULL, ...)
```

Arguments

`x`	A `tseLCA` object returned by `three_step()`.
`horiz`	Logical. If `TRUE`, item labels are drawn horizontally.
`clab`	Optional character vector of length T giving class labels.
`...`	Further arguments passed to `plot.multiLCA`.

Value

Called for its side effect (a base-graphics plot). Invisibly returns `NULL`.

Examples

```
d    <- generate_data(100, "high", "covariate", seed = 1)
fit_m <- three_step(d, paste0("Y", 1:6), n_classes = 3)
plot(fit_m)


# Custom class labels
plot(fit_m, clab = c("Low risk", "Mixed", "High risk"))
```

`print.tseLCA_measurement` *Print a tseLCA model object*

Description

Compact one-line or table summary printed to the console.

Usage

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

## S3 method for class 'tseLCA_covariate'
print(x, digits = 4, ...)

## S3 method for class 'tseLCA_distal'
print(x, digits = 4, ...)

## S3 method for class 'tseLCA_both'
print(x, digits = 4, ...)
```

Arguments

`x`	A `tseLCA` object returned by `three_step()`.
`...`	Further arguments.
`digits`	Integer. Number of decimal places for coefficient tables.

Value

Invisibly returns `x`.

Examples

```
d    <- generate_data(100, "high", "covariate", seed = 1)
fit_m <- three_step(d, paste0("Y", 1:6), n_classes = 3)
print(fit_m)

d   <- generate_data(200, "high", "covariate", seed = 1)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", use.simple.cov = TRUE)
print(fit)


d   <- generate_data(200, "high", "distal", seed = 2)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zo.name = "Zo", use.simple.cov = TRUE)
print(fit)
```

summary.tseLCA_measurement *Summarize a tseLCA model object*

Description

Verbose summary including model fit, class prevalences, item-response probabilities, and coefficient tables with standard errors and p-values.

Usage

```
## S3 method for class 'tseLCA_measurement'
summary(object, ...)

## S3 method for class 'tseLCA_covariate'
summary(object, digits = 4, ...)

## S3 method for class 'tseLCA_distal'
summary(object, digits = 4, ...)

## S3 method for class 'tseLCA_both'
summary(object, digits = 4, ...)
```

Arguments

object	A tseLCA object returned by three_step().
...	Further arguments (currently unused).
digits	Integer. Number of decimal places for coefficient tables.

Value

Invisibly returns object.

Examples

```
d    <- generate_data(100, "high", "covariate", seed = 1)
fit_m <- three_step(d, paste0("Y", 1:6), n_classes = 3)
summary(fit_m)

d   <- generate_data(200, "high", "covariate", seed = 1)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", use.simple.cov = TRUE)
summary(fit)


d   <- generate_data(200, "high", "distal", seed = 2)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zo.name = "Zo", use.simple.cov = TRUE)
```

```
summary(fit)
```

---

three_step *Three-step LCA estimation with covariates and/or distal outcomes*

Description

Fits a three-step latent class model through the following steps:

1. **Measurement model**: estimates latent class parameters (π, ϕ) using **multilevLCA** (Lyrvall et al., 2025).
2. **Classification-error matrix**: computes posterior class probabilities and the T x T misclassification probability matrix $P(W = s \mid X = t)$, with standard errors corrected for classification-error propagation (Bakk, Oberski & Vermunt, 2014).
3. **Structural model**: estimates covariate effects using two-step starting values (Bakk & Kuha, 2018) and/or distal outcome means following Bakk, Tekle & Vermunt (2013), with the ML correction (Vermunt, 2010) or BCH correction (Bolck, Croon & Hagenaars, 2004). See `vignette("tseLCA", package = "tseLCA")` for a worked example.

Usage

```
three_step(
  data,
  Y.names,
  n_classes,
  Zp.names = NULL,
  Zo.name = NULL,
  step1 = NULL,
  use.two.step = TRUE,
  use.modal.assignment = TRUE,
  include.intercept = TRUE,
  use.simple.cov = FALSE,
  incomplete = FALSE,
  boundary.tol = 0.01,
  maxIter.measurement = 5000,
  measurement.tol = 1e-08,
  covariate.tol = 1e-06,
  iter.measurement = 10L,
  R2.threshold = 0.7,
  use.bch = FALSE,
  em.maxIter = 200L,
  get.twostep.vcov = FALSE,
  rebase = "C1",
  family = "gaussian",
  correct.spec = FALSE,
  verbose = FALSE
)
```

Arguments

data	A data.frame containing all columns referenced by Y.names, Zp.names, and Zo.name.
Y.names	Character vector of indicator column names. Need to be coded as consecutive integers with base level starting at 0.
n_classes	Integer. Number of latent classes.
Zp.names	Character vector of covariate column names, or NULL for a measurement-only fit. Default NULL.
Zo.name	Single character name of the distal outcome column, or NULL. Default NULL.
step1	Pre-fitted Step-1 object (output of lca_step1() or a prior three_step() call), or NULL to run Step 1 internally. Default NULL.
use.two.step	Logical. Initialize Step-3 from two-step estimates. Default TRUE.
use.modal.assignment	Logical. Use modal (hard) class assignments in Step 2 and 3. FALSE uses soft posterior weights. Default TRUE.
include.intercept	Logical. Prepend an intercept column to the covariate design matrix. Default TRUE.
use.simple.cov	Logical. Skip the Step-1 measurement-uncertainty correction and return only the robust sandwich variance. Faster but underestimates standard errors when class separation is low. Default FALSE.
incomplete	Logical. FIML for partially missing indicators. See the Missing Data section of vignette("tseLCA", package = "tseLCA"). Default FALSE.
boundary.tol	Scalar. Parameters within this tolerance of 0 or 1 are treated as fixed when computing the Step-1 variance matrix for numerical stability. Default 1e-2.
maxIter.measurement	Integer. Maximum EM iterations for Step 1. Default 5000L.
measurement.tol	Scalar. Convergence tolerance for the Step-1 EM algorithm. Default 1e-8.
covariate.tol	Scalar. Convergence tolerance for the Step-3 Newton-Raphson or EM algorithm. Default 1e-6.
iter.measurement	Integer. Number of random restarts triggered when the Step-1 entropy R^2 falls below R2.threshold. Default 10L.
R2.threshold	Scalar. Entropy R^2 threshold below which Step-1 random restarts are triggered. Default 0.70.
use.bch	Logical. Use BCH-corrected weights instead of the ML estimator in Step 3. May error if BCH weights induce a non-positive semi-definite Hessian in the third step (common in cases of low separation). Default FALSE.
em.maxIter	Integer. Maximum EM iterations for the Step-3 covariate or distal outcome model. Default 200L.

`get.twostep.vcov`
: Logical. If `TRUE`, obtain **multilevLCA**'s bias-corrected variance-covariance matrix for the two-step gamma estimates and store it in `$two_step_vcov`. If the `fitZ` object passed via `step1` already contains a `Varmat_cor` (from a prior `fitZ_from_multiLCA()` or plain `multiLCA` call), it is attached automatically even when `get.twostep.vcov = FALSE`. Default `FALSE`.

`rebase`
: Character (e.g. `"C1"`, `"C2"`) or integer specifying which latent class to use as the reference category in the multinomial logit. The measurement model is permuted so this class becomes column 1 before any structural estimation. Default `"C1"`.

`family`
: Character. Distal outcome family: one of `"gaussian"` (class means), `"poisson"` (log-rates), or `"binomial"` (logits). Default `"gaussian"`.

`correct.spec`
: Logical. Use the model-robust outer-product Hessian for Step-3 standard errors rather than the observed-data Hessian. Not appropriate when the Step-3 model may be misspecified. Default `FALSE`.

`verbose`
: Logical. Print convergence messages. Default `FALSE`.

Value

An S3 object of class `tseLCA`. The subclass depends on which models were estimated:

`tseLCA_measurement` Returned when neither `Zp.names` nor `Zo.name` is supplied. Contains the following elements:

- `measurement_model` Step-1 output list from `lca_step1()`.
- `llik` Final Step-1 log-likelihood.
- `AIC`, `BIC` Information criteria from the measurement model.
- `R2entr` Entropy R^2 of the measurement model.
- `n_classes` Number of latent classes.
- `posteriors` N x T matrix of soft posterior class probabilities.
- `classifications` Length-N integer vector of modal class assignments.

`tseLCA_covariate` Returned when `Zp.names` is supplied and `Zo.name` is `NULL`. Contains all elements of `tseLCA_measurement` plus:

- `three_step` Q x (T-1) matrix of Step-3 gamma coefficients.
- `three_step_vcov` Q(T-1) x Q(T-1) variance-covariance matrix for `three_step`, with measurement-uncertainty correction unless `use.simple.cov = TRUE`.
- `two_step` Q x (T-1) matrix of two-step starting values, or `NULL` if `use.two.step = FALSE`.
- `two_step_vcov` **multilevLCA** bias-corrected vcov for the two-step estimates, or `NULL`.
- `estimator` Character: `"ML"` or `"BCH"`.
- `entropy.R2` Covariate-adjusted entropy R^2.
- `llik` Profile log-likelihood $\sum_i \log \sum_t P(X = t|Z_{p,i};\hat{\gamma})P(Y_i|X = t;\hat{\phi})$, with Step-1 parameters $\hat{\phi}$ held fixed. By construction smaller than the equivalent one-step MLE likelihood.

`tseLCA_distal` Returned when `Zo.name` is supplied and `Zp.names` is `NULL`. Contains:

- `three_step` Named length-T vector of Step-3 distal outcome parameters (means, log-rates, or logits depending on `family`).

`three_step_vcov` T x T variance-covariance matrix for `three_step`, named `mu_C1` through `mu_CT`.

`three_step.llik` Step-3 distal log-likelihood $\log P(Z_o|X=t)$ at converged estimates.

`llik` Profile log-likelihood $\sum_i \log \sum_t P(X=t|\hat{\pi})P(Z_{o,i}|X=t;\hat{\mu})P(Y_i|X=t;\hat{\phi})$, with Step-1 parameters $\hat{\pi}, \hat{\phi}$ held fixed. By construction smaller than the equivalent one-step MLE likelihood.

`AIC` Akaike information criterion based on `llik`.

`BIC` Bayesian information criterion based on `llik`, using the number of distal-complete observations.

`family` Character. The distal outcome family used.

`estimator` Character: `"ML"` or `"BCH"`.

`posteriors` N x T soft posterior matrix.

`classifications` Length-N modal class assignment vector.

`tseLCA_both` Returned when both `Zp.names` and `Zo.name` are supplied. Contains:

`covariate` A `tseLCA_covariate`-structured sub-list (see above), including `llik`, `AIC`, `BIC`, `entropy.R2`.

`distal` A `tseLCA_distal`-structured sub-list (see above), including `llik`, `AIC`, `BIC`, `three_step.llik`.

`family`, `n_classes`, `estimator` Shared top-level fields.

`posteriors`, `classifications` Shared N x T posterior matrix and length-N modal class vector.

References

Bakk, Z., Tekle, F. B., & Vermunt, J. K. (2013). Estimating the association between latent class membership and external variables using bias-adjusted three-step approaches. *Sociological Methodology*, 43(1), 272–311. doi:10.1177/0081175012470644

Bakk, Z., & Kuha, J. (2018). Two-step estimation of models between latent classes and external variables. *Psychometrika*, 83(4), 871–892. doi:10.1007/s1133601795927

Bakk, Z., Pohle, M. J., & Kuha, J. (2025). Bias-adjusted three-step estimation of structural models for latent classes. *Multivariate Behavioral Research*. doi:10.1080/00273171.2025.2473935

See Also

`vignette("tseLCA", package = "tseLCA")` for a full worked example; `lca_step1()` for standalone Step-1 estimation; `fitZ_from_fit0()` and `fitZ_from_multiLCA()` for two-step covariate estimation.

Examples

```
d <- generate_data(n = 200, separation = "high",
                   scenario = "covariate", seed = 1)

# Measurement model only
fit_m <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3)
summary(fit_m)

# ML three-step with simple SEs (fast)
```

```
fit <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", use.simple.cov = TRUE)
summary(fit)
coef(fit)
vcov(fit)

# Full measurement-uncertainty correction (see vignette for interpretation)
fit_cor <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                      Zp.names = "Zp", use.simple.cov = FALSE,
                      use.modal.assignment = FALSE)
summary(fit_cor)

# BCH estimator
fit_bch <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                      Zp.names = "Zp", use.bch = TRUE,
                      use.simple.cov = TRUE)
summary(fit_bch)

# Change reference class
fit_c2 <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                     Zp.names = "Zp", use.simple.cov = TRUE,
                     rebase = "C2")
summary(fit_c2)

# Gaussian distal outcome
d2 <- generate_data(200, "high", "distal", seed = 2)
fit_dis <- three_step(d2, Y.names = paste0("Y", 1:6), n_classes = 3,
                      Zo.name = "Zo", family = "gaussian",
                      use.simple.cov = TRUE)
summary(fit_dis)

# Pass a pre-fitted measurement model to skip Step 1
fit_step1 <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3)
fit2 <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                   Zp.names = "Zp", step1 = fit_step1,
                   use.simple.cov = TRUE)
summary(fit2)

# Plot item-response profiles from the measurement model
plot(fit)
```

`vcov.tseLCA_measurement` *Extract the variance-covariance matrix from a tseLCA model object*

Description

For measurement models, returns the BHHH variance-covariance matrix in the unconstrained log-ratio parameterization (NOT the probability scale). Row and column names identify each parameter

as `log(pi_t/pi_1)` (class prevalences) or `log(P(Y=k|C_t)/P(Y=0|C_t))` (item-response probabilities). An attribute `"parameterization"` is attached to remind the user of the scale.

Usage

```
## S3 method for class 'tseLCA_measurement'
vcov(object, boundary.tol = 0.01, ...)

## S3 method for class 'tseLCA_covariate'
vcov(object, which = c("three_step", "two_step"), ...)

## S3 method for class 'tseLCA_distal'
vcov(object, ...)

## S3 method for class 'tseLCA_both'
vcov(
  object,
  which = c("both", "covariate", "distal"),
  step = c("three_step", "two_step"),
  ...
)
```

Arguments

`object`	A `tseLCA` object returned by `three_step()`.
`boundary.tol`	Scalar. Parameters within this tolerance of 0 or 1 are treated as fixed. Default `1e-2`.
`...`	Further arguments (currently unused).
`which`	Character. `"three_step"` (default) or `"two_step"` for covariate models; `"covariate"`, `"distal"`, or `"both"` for both models.
`step`	Character. For `tseLCA_both`: `"three_step"` (default) or `"two_step"`.

Value

A named square matrix in the unconstrained log-ratio parameterization. Row/column names identify each parameter as `log(pi_t/pi_1)` or `log(P(Y=k|C_t)/P(Y=0|C_t))`. An attribute `"parameterization"` is attached as a reminder. Returns `NULL` invisibly if `fit0$mU` is not available. For structural models, returns the Step-3 vcov matrix; the two-step vcov is only available when `get.twostep.vcov = TRUE`.

Examples

```
d    <- generate_data(100, "high", "covariate", seed = 1)
fit_m <- three_step(d, paste0("Y", 1:6), n_classes = 3)
V <- vcov(fit_m)
# Names show log-ratio parameterization:
rownames(V)
attr(V, "parameterization")
```

```
d   <- generate_data(200, "high", "covariate", seed = 1)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", use.simple.cov = TRUE)
vcov(fit)   # Q*(T-1) x Q*(T-1) vcov matrix with named rows/cols


d   <- generate_data(200, "high", "distal", seed = 2)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zo.name = "Zo", use.simple.cov = TRUE)
vcov(fit)   # T x T vcov matrix with mu_C1..mu_CT row/col names


d   <- generate_data(200, "high", "covariate", seed = 1)
d$Zo <- rnorm(200, mean = c(-1, 0, 1)[d$X], sd = 0.5)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zp.names = "Zp", Zo.name = "Zo",
                  use.simple.cov = TRUE)
vcov(fit, which = "covariate")
vcov(fit, which = "distal")
```

Index