Package: tseLCA (via r-universe)

October 31, 2026

Type Package

Title Three-Step Estimation for Latent Class Analysis

Version 2.1.1

Description Bias-adjusted three-step estimation of latent class models with covariates and distal outcomes. The latent class measurement model is estimated first, with 'multilevLCA' (Lyrvall et al., 2025) <doi:10.1080/00273171.2025.2473935>, and held fixed; observations are then classified; and the classes are related to covariates and distal outcomes with the maximum likelihood correction of Vermunt (2010) <doi:10.1093/pan/mpq025> and Bakk, Tekle and Vermunt (2013) <doi:10.1177/0081175012470644>, or the correction of Bolck, Croon and Hagenaars (2004) <doi:10.1093/pan/mph001>. Standard errors account for the uncertainty of the measurement model (Bakk, Oberski and Vermunt, 2014) <doi:10.1093/pan/mpu003>. Includes class enumeration, modal and proportional class assignment, covariate formulas, Gaussian, Poisson, binomial, and multinomial distal outcomes, the two-step estimator of Bakk and Kuha (2018) <doi:10.1007/s11336-017-9592-7>, measurement models applied to new samples, and full-information maximum likelihood for missing indicators, standard methods for fitted models, and a data-generating process replicating the simulation design of Bakk and Kuha (2018).

License GPL (>= 3)

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports cli, Formula, multilevLCA

Suggests nnet, 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

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-10-31 21:05:15 UTC

RemoteUrl https://github.com/samleebyu/tselca

RemoteRef HEAD

RemoteSha 00eb8b65238e655f21aacf87628039712d1b8d67

Contents

anova.tseLCA_covariate . . . 3
as_tse_lca . . . 4
best_model . . . 5
bk2018_params . . . 6
class_sizes . . . 7
coef.tseLCA_structural . . . 8
draw_classes . . . 10
draw_classes_given_Zp . . . 10
draw_indicators . . . 11
draw_Zo . . . 11
draw_Zp . . . 12
fitZ_from_fit0 . . . 12
fitZ_from_multiLCA . . . 14
generate_all_conditions . . . 16
generate_data . . . 17
lca_step1 . . . 19
logLik.tseLCA . . . 21
make_rho . . . 22
measurement . . . 23
mnl_probs . . . 24
omnibus_test . . . 25
plot.tseLCA . . . 26
posterior . . . 26
predict.tseLCA_covariate . . . 27
predict.tseLCA_measurement . . . 28
relevel.tseLCA_covariate . . . 29
summary.tseLCA_structural . . . 30
three_step . . . 31
tse_classify . . . 38
tse_control . . . 39
tse_covariate . . . 40

tse_distal 42
tse_lca 43
tse_twostep 45
tseLCA 46
vcov.tseLCA_structural 48

Index **50**

`anova.tseLCA_covariate` *Wald tests of covariate terms*

Description

Tests, term by term, that all multinomial-logit coefficients of a covariate term are zero (for all non-reference classes), using the model's variance matrix.

Usage

```
## S3 method for class 'tseLCA_covariate'
anova(object, ...)
```

Arguments

`object`	A `tseLCA_covariate` object from `tse_covariate()`.
`...`	Unused.

Value

An anova table with the Wald statistic, degrees of freedom, and p-value of each term.

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
anova(tse_covariate(tse_classify(m), ~ Zp))
```

`as_tse_lca` *Use a measurement model with given parameters (Step 1)*

Description

Creates a measurement model from given class sizes and item-response probabilities, evaluated on `data`, without estimating it. This allows Steps 2 and 3 to be based on a measurement model estimated elsewhere: in another program, reported in a publication, or saved from an earlier analysis.

Usage

```
as_tse_lca(
  formula,
  data,
  class_sizes,
  item_probs,
  vcov = NULL,
  missing = c("listwise", "fiml"),
  control = tse_control()
)
```

Arguments

`formula`	`cbind(Y1, Y2, ...) ~ 1`, as in `tse_lca()`.
`data`	A data frame.
`class_sizes`	Class proportions, one per class (they are normalized to sum to one).
`item_probs`	Item-response probabilities in the layout of `item_probs()`: one column per class, and one row per binary item ($P(Y = 1 \mid X = t)$, where 1 is the item's second category) or per category of a polytomous item ($P(Y = k \mid X = t)$), in the order of the indicators.
`vcov`	Optional variance matrix of the Step-1 parameters, for corrected standard errors in Step 3. It refers to the unconstrained log-ratio parameters, in the layout of `vcov()` of a `tse_lca()` model: the class-size log-ratios $\log(\pi_t/\pi_1)$, $t = 2, \ldots, T$, then, class by class and item by item, $\log(P(Y = k \mid X = t)/P(Y = 0 \mid X = t))$ for each category $k \geq 1$. Classes and items are in the order of `class_sizes` and of the indicators. Row and column names are ignored.
`missing, control`	As in `tse_lca()`.

Details

Corrected standard errors in Step 3 (`se = "corrected"`) need the variance matrix of the Step-1 parameters, which can only come from the sample they were estimated on. A model from `as_tse_lca()` receives only the parameter values, so it uses the variance matrix given in `vcov`.

Without vcov, Step 3 warns and reports robust standard errors (se = "robust"), which do not include the Step-1 uncertainty. vcov() of the model returns vcov, or an error if none was given.

If the parameters are the maximum likelihood estimates for data (a model estimated on these data and saved), vcov(m) of the original tse_lca() model is the matching vcov. For parameters estimated on another sample, take the variance matrix from that estimation (e.g. Latent GOLD or Mplus output), or use tse_classify() with newdata on the original model.

Value

A tseLCA_measurement object, usable like one from tse_lca().

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)

# the same measurement model from its parameters and their variance
m2 <- as_tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d,
                 class_sizes = class_sizes(m), item_probs = item_probs(m),
                 vcov = vcov(m))
all.equal(logLik(m2), logLik(m), tolerance = 1e-6)
coef(tse_covariate(tse_classify(m2), ~ Zp))

# without `vcov`, Step 3 warns and reports robust standard errors
m3 <- as_tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d,
                 class_sizes = class_sizes(m), item_probs = item_probs(m))
fit3 <- tse_covariate(tse_classify(m3), ~ Zp)
```

---

best_model *Class enumeration results*

Description

Methods for the tseLCA_select object returned by tse_lca() with several numbers of classes. best_model() returns the fitted model that minimizes an information criterion; x[[k]] returns the k-class model.

Usage

```
best_model(object, ...)

## S3 method for class 'tseLCA_select'
best_model(object, criterion = c("BIC", "AIC", "SABIC"), ...)

## S3 method for class 'tseLCA_select'
x[[i, ...]]

## S3 method for class 'tseLCA_select'
```

```
as.data.frame(x, ...)

## S3 method for class 'tseLCA_select'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'tseLCA_select'
plot(x, which = c("AIC", "BIC", "SABIC"), ...)
```

Arguments

`...`	Further arguments passed to `graphics::matplot()` (`plot`) or unused.
`criterion`	Information criterion to minimize: `"BIC"` (default), `"AIC"`, or `"SABIC"`.
`x, object`	A `tseLCA_select` object.
`i`	Number of classes of the model to extract.
`digits`	Number of significant digits to print.
`which`	Criteria to plot.

Value

`best_model()` and `[[`: a `tseLCA_measurement` object. `as.data.frame()`: the enumeration table. `print()`, `plot()`: `x`, invisibly.

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
sel <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 1:4)
as.data.frame(sel)
best_model(sel)
sel[[2]]
```

---

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
```

`class_sizes` *Class sizes and item-response probabilities of the measurement model*

Description

`class_sizes()` returns the estimated class proportions and `item_probs()` the class-conditional item-response probabilities of the Step-1 measurement model underlying any fitted `tseLCA` object.

Usage

```
class_sizes(object, ...)

## S3 method for class 'tseLCA'
class_sizes(object, se = FALSE, ...)

item_probs(object, ...)
```

```
## S3 method for class 'tseLCA'
item_probs(object, se = FALSE, ...)
```

Arguments

`object`	A fitted `tseLCA` object.
`...`	Further arguments (currently unused).
`se`	Logical. If `TRUE`, also return standard errors.

Details

With `se = TRUE`, their standard errors are returned as well. They are obtained by the delta method from the variance of the measurement model's log-ratio parameters (`vcov()` of the measurement model): class sizes are the softmax of $\log(\pi_t/\pi_1)$, and the response probabilities of an item in class t the softmax of $\log(P(Y = k \mid X = t)/P(Y = 0 \mid X = t))$. Parameters on the boundary of the parameter space are treated as fixed and get a standard error of zero.

Value

`class_sizes()`: a named numeric vector of length T summing to one. `item_probs()`: a matrix with one row per item (binary items: $P(Y = 1 \mid X = t)$) or per item category (polytomous items: $P(Y = k \mid X = t)$) and one column per class. With `se = TRUE`, a list with elements `estimate` and `se` of that form.

Examples

```
d <- generate_data(200, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
class_sizes(m)
item_probs(m)
item_probs(m, se = TRUE)$se
```

---

coef.tseLCA_structural

Coefficients of a fitted tseLCA model

Description

Returns a named coefficient vector whose names match the rows and columns of `vcov()`, so that `stats::confint()` and other generic tools work.

Usage

```
## S3 method for class 'tseLCA_structural'
coef(
  object,
  component = c("all", "covariate", "distal"),
  step = c("three_step", "two_step"),
  matrix = FALSE,
  ...
)

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

Arguments

`object`	A fitted `tseLCA` object.
`component`	For `tseLCA_both` objects: `"all"` (default; covariate then distal coefficients), `"covariate"`, or `"distal"`.
`step`	`"three_step"` (default) or `"two_step"` (the two-step estimates used to initialize Step 3; covariate models only).
`matrix`	Logical. If `TRUE`, return the coefficients in their natural matrix layout: (Q+1) x (T-1) for covariate models, T x C for multinomial distal outcomes, a list of both for `tseLCA_both`.
`...`	Further arguments (currently unused).

Details

- Measurement models: class-size log-ratios $\log(\pi_t/\pi_1)$ and item-response log-ratios $\log(P(Y = k \mid t)/P(Y = 0 \mid t))$. Use `class_sizes()` and `item_probs()` for the probability scale.
- Covariate models: multinomial-logit coefficients named `"<covariate>:<class>"`, e.g. `"Zp:C2"`.
- Distal-outcome models: class-specific means (`gaussian`), log means (`poisson`), logits (`binomial`), or category probabilities (`multinomial`, named `"<class>:<category>"`).

Value

A named numeric vector (or matrix / list if `matrix = TRUE`).

Examples

```
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)
coef(fit, matrix = TRUE)
confint(fit)
```

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)) with an EM algorithm using 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,
  Y.levels = NULL,
  Zp.formula = NULL
)
```

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+1) x (T-1) starting value matrix for mGamma.
verbose	Logical. Print convergence messages. Default FALSE.
Y.levels	Optional named list of indicator categories when data holds 0-based codes (as stored with a fitted measurement model); NULL derives them from data.
Zp.formula	Optional one-sided formula for the covariate design; it replaces Zp.names and include.intercept.

Value

A list with the following elements:

mGamma (Q+1) x (T-1) numeric matrix of multinomial logit coefficients, where Q + 1 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.

Deprecated

Deprecated as of tseLCA 2.0.0: use `tse_twostep()`. It keeps working and warns once per session when called directly.

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+1) x (T-1) coefficient matrix
fZ$converged
```

---

fitZ_from_multiLCA *Estimate two-step covariate model with 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",
  startval = NULL,
  n_init = NULL,
  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".
startval	Optional starting classification for the measurement portion of this multiLCA(fixedpars = 1) fit – an integer vector or a conditional item-response probability matrix, as described in lca_step1_startval() – e.g. the same value passed to lca_step1() for the primary Step-1 fit. When supplied, kmea = FALSE is used and iter.measurement/R2.threshold restarts are skipped, for the same reasons as in lca_step1_startval(). Mutually exclusive with n_init. Default NULL.
n_init	Optional positive integer. If supplied, fits this multiLCA(fixedpars = 1) model n_init times from independent uniform-random classifications (kmea = FALSE) and keeps the fit with the highest log-likelihood, as in lca_step1()'s n_init argument. iter.measurement/R2.threshold restarts are skipped. Mutually exclusive with startval. Default NULL.
verbose	Logical.

Value

A list with the following elements:

`mGamma` (Q+1) 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.

Deprecated

Deprecated as of tseLCA 2.0.0: use `tse_twostep()`. It keeps working and warns once per session when called directly.

Examples

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

# Two-step estimation with 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 with multinomial logit. "distal" Latent X predicts continuous Zo with 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",
  startval = NULL,
  n_init = NULL,
  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.

startval — Optional starting classification for the Step-1 measurement model: either an integer vector of length nrow(data) (1..n_classes per row) or a numeric matrix of conditional item-response probabilities from which a classification is derived. See lca_step1_startval() for the full description of both forms. When supplied, lca_step1() fits the measurement model with lca_step1_startval(), not multilevLCA's default k-means-on-principal-components initialization, and estimate.one.step, iter.measurement, and R2.threshold (which govern the default restart-on-low-entropy behavior) are ignored. Mutually exclusive with n_init. Default NULL.

n_init — Optional positive integer. If supplied, fits the measurement model n_init times from independent uniform-random classifications (each through startval-style injection with kmea = FALSE, not multilevLCA's k-means-on-PCA path) and keeps the fit with the highest log-likelihood – the unconditional multi-start analog of n_init in **StepMix** or nrep in **poLCA**. Unlike iter.measurement (which re-runs multilevLCA's own k-means initialization, and only when entropy R^2 is low), all n_init fits are always run. estimate.one.step, iter.measurement, and R2.threshold are ignored when n_init is supplied. Mutually exclusive with startval. Default NULL.

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).

Deprecated

Deprecated as of tseLCA 2.0.0: use tse_lca(). It keeps working and warns once per session when called directly.

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

# Many random-classification restarts, keeping the best (analogous to
# n_init in StepMix or nrep in poLCA)
s1r <- lca_step1(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                 n_init = 20L, verbose = TRUE)
```

`logLik.tseLCA` *Log-likelihood, number of observations, and information criteria*

Description

`logLik()` returns the log-likelihood of a fitted model with its number of free parameters (`df`) and observations (`nobs`), so that `stats::AIC()` and `stats::BIC()` work directly. For a measurement model this is the Step-1 log-likelihood. For structural models it is the log-likelihood of the joint model for the indicators and the structural variables evaluated with the Step-1 parameters held fixed; for a `tseLCA_both` object it is the distal-outcome component, which conditions on the covariates.

Usage

```
## S3 method for class 'tseLCA'
logLik(object, ...)

## S3 method for class 'tseLCA'
nobs(object, ...)
```

Arguments

`object`	A fitted `tseLCA` object.
`...`	Further arguments (currently unused).

Value

`logLik()`: an object of class `"logLik"`. `nobs()`: an integer.

Examples

```
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)
logLik(fit)
AIC(fit)
BIC(fit)
nobs(fit)
```

`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)
```

`measurement` *Components of a fitted tseLCA model*

Description

Extract the step-wise components of a model fitted with `tseLCA()` or the step-wise functions: the Step-1 measurement model, the Step-2 classification, and the Step-3 covariate and distal outcome models.

Usage

```
measurement(x, ...)

## S3 method for class 'tseLCA'
measurement(x, ...)

classification(x, ...)

## S3 method for class 'tseLCA'
classification(x, ...)

covariate(x, ...)

## S3 method for class 'tseLCA'
covariate(x, ...)

distal(x, ...)

## S3 method for class 'tseLCA'
distal(x, ...)
```

Arguments

`x`	A fitted tseLCA object.
`...`	Unused.

Value

`measurement()`: a `tseLCA_measurement` object. `classification()`: a `tseLCA_classify` object. `covariate()`: a `tseLCA_covariate` object. `distal()`: a `tseLCA_distal` object.

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
d$Zo <- draw_Zo(d$X, bk2018_params$distal_params)
fit <- tseLCA(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ Zp | Zo, data = d, nclass = 3)
class_sizes(measurement(fit))
```

```
classification(fit)$D
coef(covariate(fit))
omnibus_test(distal(fit))
```

`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)
```

omnibus_test *Omnibus Wald test of class equality for a distal outcome*

Description

Tests whether the distal outcome distribution differs across latent classes at all – $H_0 : \theta_1 = \theta_2 = \cdots = \theta_T$ for the class-specific distal parameters θ_t (class means for `family = "gaussian"`, log-rates for `"poisson"`, logits for `"binomial"`, or the full length-C category-probability vector for `"multinomial"`) – using a generalized Wald test with `vcov()`. This answers whether the outcome's distribution is associated with class membership at all, before drilling into which classes differ. The degrees of freedom equal the rank of the contrast covariance (`T - 1` for scalar outcomes; `(T - 1) * (C - 1)` for multinomial, i.e. the textbook chi-squared test of homogeneity in a $T \times C$ table), computed with a Moore-Penrose pseudo-inverse so the test remains valid despite `multinomial`'s inherently singular covariance (each class's category probabilities sum to 1).

Usage

```
omnibus_test(object, ...)

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

## S3 method for class 'tseLCA_both'
omnibus_test(object, ...)
```

Arguments

`object`	A `tseLCA_distal` object, or a `tseLCA_both` object (tests its distal component).
`...`	Unused; present for S3 method consistency.

Value

A standard `"htest"` object: the Wald chi-squared `$statistic`, its degrees of freedom `$parameter` (also `$df`), and the `$p.value`.

Examples

```
d <- generate_data(300, "high", "distal", seed = 1)
fit <- three_step(d, paste0("Y", 1:6), n_classes = 3,
                  Zo.name = "Zo", use.simple.cov = TRUE)
omnibus_test(fit)
```

plot.tseLCA *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 of the Step-1 measurement model.

Usage

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

Arguments

`x`	A fitted `tseLCA` object.
`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)
plot(fit_m, clab = c("Low", "Mixed", "High"))
```

posterior *Posterior class-membership probabilities and modal class assignments*

Description

`posterior()` returns the n x T matrix of posterior class-membership probabilities used by a fitted model; `classes()` returns the modal (most likely) class of each observation.

Usage

```
posterior(object, ...)

## S3 method for class 'tseLCA'
posterior(object, ...)

classes(object, ...)

## S3 method for class 'tseLCA'
classes(object, ...)
```

Arguments

`object`	A fitted `tseLCA` object.
`...`	Further arguments (currently unused).

Value

`posterior()`: a numeric n x T matrix. `classes()`: an integer vector of length n with values in `1..T`.

Examples

```
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)
head(posterior(fit))
table(classes(fit))
```

---

predict.tseLCA_covariate

Class-membership probabilities from a covariate model

Description

The fitted class prior $P(X = t \mid Z)$ for the rows of `newdata`, or of the estimation data.

Usage

```
## S3 method for class 'tseLCA_covariate'
predict(object, newdata = NULL, type = c("prob", "class"), ...)
```

Arguments

object	A tseLCA_covariate object from tse_covariate().
newdata	Optional data frame with the covariates.
type	"prob" (default) for the n x T probability matrix, or "class" for the most likely class.
...	Unused.

Value

A matrix (rows with missing covariates are NA) or integer vector.

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
fit <- tse_covariate(tse_classify(m), ~ Zp)
predict(fit, newdata = data.frame(Zp = 1:5))
```

---

predict.tseLCA_measurement

Class membership predictions from a measurement model

Description

Posterior class-membership probabilities P(X = t | Y) for the rows of newdata (or the estimation sample), or their modal class. Indicators in newdata are coded with the categories stored in the model; missing indicator values are skipped (the posterior uses the observed ones), and rows with no observed indicator get NA.

Usage

```
## S3 method for class 'tseLCA_measurement'
predict(object, newdata = NULL, type = c("posterior", "class"), ...)

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

Arguments

object	A tseLCA_measurement object.
newdata	Optional data frame with the indicator columns. Omitted: the estimation sample.
type	"posterior" (default) for an n x T matrix of probabilities, or "class" for the modal class of each row.
...	Unused.

Value

A matrix (`type = "posterior"`) or integer vector (`"class"`).

Examples

```
d <- generate_data(300, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
predict(m, newdata = d[1:5, ])
predict(m, newdata = d[1:5, ], type = "class")
```

---

relevel.tseLCA_covariate

Change the reference class of a covariate model

Description

Refits the model with another reference class of the multinomial logit.

Usage

```
## S3 method for class 'tseLCA_covariate'
relevel(x, ref, ...)
```

Arguments

`x`	A `tseLCA_covariate` object from `tse_covariate()`.
`ref`	The new reference class (number or label such as `"C2"`).
`...`	Unused.

Value

The refitted `tseLCA_covariate` object.

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
fit <- tse_covariate(tse_classify(m), ~ Zp)
coef(stats::relevel(fit, ref = "C3"))
```

summary.tseLCA_structural *Summarize a fitted tseLCA model*

Description

summary() collects fit statistics and coefficient tables; printing the result formats the tables with stats::printCoefmat(). The coefficient table (columns Estimate, Std. Error, z value, Pr(>|z|)) can be extracted with coef(summary(fit)).

Usage

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

## S3 method for class 'summary.tseLCA_structural'
coef(object, ...)

## S3 method for class 'summary.tseLCA_structural'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

## S3 method for class 'tseLCA_structural'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)

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

## S3 method for class 'summary.tseLCA_measurement'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

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

Arguments

object	A fitted tseLCA object.
...	Further arguments passed to stats::printCoefmat().

x	A summary.tseLCA_structural or summary.tseLCA_measurement object, or a fitted tseLCA object (for print).
digits	Number of significant digits to print.
signif.stars	Logical; print significance stars?

Value

summary() returns an object of class "summary.tseLCA_structural" or "summary.tseLCA_measurement". Print methods return their argument invisibly.

Examples

```
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)
printCoefmat(coef(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,
  startval = NULL,
  n_init = 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. Indicators may be factors, logicals, character, or numeric codes (see `tse_lca()`).
`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`.
`startval`	Optional starting classification for the Step-1 measurement model, either an integer vector of length `nrow(data)` (a class assignment `1..n_classes` for every row) or a numeric matrix of conditional item-response probabilities $P(Y_h = k \mid X = t)$ (one row per item-category pair in `Y.names` order, one column per class) from which a classification is derived internally. See `lca_step1_startval()` for the full description of both forms and typical sources (an external solver run with many random starts, or a published item-response table). **multilevLCA**'s default initialization (k-means on principal components) is deterministic given the data and can converge to a local optimum of the Step-1 log-likelihood; supplying `startval` bypasses it (`kmea = FALSE` with the classification injected as multilevLCA's `startval`). Mutually exclusive with `step1` and `n_init`. Default `NULL`.
`n_init`	Optional positive integer. If supplied, fits the Step-1 measurement model `n_init` times from independent uniform-random classifications (`kmea = FALSE`, not multilevLCA's k-means-on-PCA path) and keeps the fit with the highest log-likelihood – the unconditional multi-start analog of `n_init` in **StepMix** or `nrep` in **poLCA**.

This is distinct from `iter.measurement`, which reruns multilevLCA's own k-means initialization, and only when the entropy R^2 of the default fit is below `R2.threshold`; `n_init` restarts always run. Mutually exclusive with `step1` and `startval`. 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 the BCH estimator in Step 3 (default: the ML estimator). 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 through `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), `"binomial"` (logits), or `"multinomial"` (a saturated model for a nominal categorical outcome with 2 or more categories – `Zo.name` may be a factor, character, or integer column; categories are taken from `sort(unique(data[[Zo.name]]))` with `factor()`). For `"multinomial"`, `coef()` returns a `T` x `C` matrix of class-conditional category probabilities $\hat{\pi}_{tc} = P(Zo = c \mid X = t)$ (rows sum to 1), not a length-`T` vector, and `vcov()` returns its `(T*C)` x `(T*C)` sandwich covariance (necessarily singular, since each class's row sums to 1 – see `omnibus_test()` for a Wald test that accounts for this). Unlike `"binomial"`, whose `coef()`/`vcov()` are on the logit scale, `"multinomial"` reports `coef()`/`vcov()` directly on the probability scale, so `Std.Error` is directly interpretable without a delta-method back-transform – but a symmetric interval `Estimate +/- 1.96*Std.Error` can fall outside $[0, 1]$ for a probability near a boundary, the same well-known limitation as a naive Wald interval for any sample proportion. The `z.value`/`p.value` columns `summary()`/`print()` show for this family test each probability against 0, which is rarely the question of interest; `omnibus_test()` is the intended, boundary-safe test of whether the outcome's distribution differs across classes. Combining `family = "multinomial"` with both `Zp.names` and `Zo.name` fully propagates both Step-1 measurement and Step-3 covariate uncertainty under `use.simple.cov = FALSE`, the same as the other families. Default `"gaussian"`.

`correct.spec` Logical. Estimate the Step-3 information matrix by the outer product of the case-wise scores, not the observed-data Hessian. Valid only when the Step-3 model is correctly specified; the default observed-Hessian sandwich is robust to misspecification. 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+1) x (T-1) matrix of Step-3 gamma coefficients.
- `three_step_vcov` (Q+1)(T-1) x (Q+1)(T-1) variance-covariance matrix for `three_step`, with measurement-uncertainty correction unless `use.simple.cov = TRUE`.
- `two_step` (Q+1) 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`) – or, for `family = "multinomial"`, a `T x C` matrix of class-conditional category probabilities (rows sum to 1).

`three_step_vcov` T x T variance-covariance matrix for `three_step`, named `mu_C1` through `mu_CT` – or, for `family = "multinomial"`, a `(T*C) x (T*C)` (necessarily rank-deficient) matrix named `"C{t}:{category}"`.

`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.

Deprecated

Deprecated as of tseLCA 2.0.0. It keeps working (and gives the same estimates) but warns once per session; set `options(tseLCA.warn.deprecated = FALSE)` to silence the warning. Use `tseLCA()` or the step-wise functions:

`three_step()`
`Y.names`, `n_classes`
`Zp.names`
`Zo.name`, `family`
`step1` (measurement model from another sample)
`startval`
`use.modal.assignment`
`use.bch`

use.simple.cov
rebase
incomplete
n_init, maxIter.measurement, measurement.tol, iter.measurement, R2.threshold, em.maxIter, covariate.tol, bou
get.twostep.vcov

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 (including from an externally supplied starting classification, with its own startval argument); 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)

# Nominal categorical distal outcome (3+ categories): coef() returns a
# T x C matrix of class-conditional category probabilities; omnibus_test()
# gives a single Wald test of whether the category distribution differs
# across classes at all.
d2$Zcat <- factor(sample(c("low", "mid", "high"), nrow(d2), replace = TRUE))
fit_cat <- three_step(d2, Y.names = paste0("Y", 1:6), n_classes = 3,
                      Zo.name = "Zcat", family = "multinomial",
                      use.simple.cov = TRUE)
coef(fit_cat)
omnibus_test(fit_cat)

# 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)

# Supply an external starting classification for Step 1 (bypasses
# multilevLCA's k-means-on-PCA initialization; here we use the DGP's own
# true classes as a stand-in for e.g. a StepMix solution with many
# random starts)
fit_ext <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                      startval = d$X, use.simple.cov = TRUE)
summary(fit_ext)

# Many random-classification restarts for Step 1, keeping the best
# (analogous to n_init in StepMix or nrep in poLCA)
fit_ninit <- three_step(d, Y.names = paste0("Y", 1:6), n_classes = 3,
                        n_init = 20L, use.simple.cov = TRUE)
summary(fit_ninit)

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

`tse_classify` *Assign observations to latent classes (Step 2)*

Description

Computes posterior class-membership probabilities from a fitted measurement model, assigns observations to classes, and estimates the classification-error probabilities $D_{ts} = P(W = s \mid X = t)$ between the true class X and the assigned class W. These error probabilities are what the bias-adjusted Step-3 estimators (BCH and ML) correct for.

Usage

```
tse_classify(
  object,
  newdata = NULL,
  assignment = c("modal", "proportional"),
  control = NULL
)

## S3 method for class 'tseLCA_classify'
print(x, digits = max(3L, getOption("digits") - 4L), ...)
```

Arguments

`object`	A measurement model from `tse_lca()` (two or more classes).
`newdata`	Optional data frame to classify. Omitted: the data the measurement model was estimated on.
`assignment`	`"modal"` (default): each observation is assigned to its most likely class. `"proportional"`: each observation is assigned to every class with its posterior probability as weight.
`control`	Estimation settings; default: those of `object`. See `tse_control()`.
`x`	A `tseLCA_classify` object.
`digits`	Number of significant digits to print.
`...`	Unused.

Details

The measurement model is held fixed. With `newdata`, observations from another sample are classified with it, e.g. to relate the classes to covariates observed only in a subsample; the uncertainty of the measurement model is then still that of the sample it was estimated on.

Value

A `tseLCA_classify` object with components `posteriors` (n x T), `classifications` (modal classes), `weights` (the assignment weights $P(W = s \mid Y_i)$), `D` (the T x T classification-error matrix), `entropy.R2` (computed from these posteriors, relative to the entropy of the class proportions), and `data`. Pass it to the Step-3 functions.

References

Vermunt, J. K. (2010). Latent class modeling with covariates: Two improved three-step approaches. *Political Analysis*, 18(4), 450–469. doi:10.1093/pan/mpq025

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
cl <- tse_classify(m)
cl
tse_classify(m, assignment = "proportional")

# classify another sample with the same measurement model
tse_classify(m, newdata = d[1:200, ])
```

`tse_control` *Estimation settings for tseLCA models*

Description

Collects the numerical settings of the three estimation steps. Pass the result as the `control` argument of the model-fitting functions.

Usage

```
tse_control(
  step1.maxit = 5000L,
  step1.tol = 1e-08,
  step1.restarts = 10L,
  step1.restart.R2 = 0.7,
  n_init = NULL,
  step3.maxit = 200L,
  step3.tol = 1e-06,
  boundary.tol = 0.01,
  hessian = c("observed", "opg"),
  verbose = FALSE
)
```

Arguments

`step1.maxit`	Maximum number of EM iterations for the Step-1 measurement model. A fit that reaches it is retried with twice as many.
`step1.tol`	Convergence tolerance of the Step-1 EM algorithm (change in log-likelihood).
`step1.restarts`	Number of additional random starts tried when the default Step-1 fit has an entropy R^2 below `step1.restart.R2`; the fit with the highest log-likelihood is kept.

`step1.restart.R2`
: Entropy R^2 threshold that triggers `step1.restarts`.

`n_init`
: Optional number of Step-1 fits from independent random classifications, bypassing the default k-means initialization; the fit with the highest log-likelihood is kept. Unlike `step1.restarts`, these always run. `NULL` (default) uses the default initialization.

`step3.maxit`
: Maximum number of iterations for the Step-3 (structural model) EM or Newton-Raphson algorithm.

`step3.tol`
: Convergence tolerance of the Step-3 algorithm.

`boundary.tol`
: Step-1 probabilities within this distance of 0 or 1 are treated as fixed when computing the Step-1 variance.

`hessian`
: Step-3 information matrix used for standard errors. `"observed"` (default) uses the analytic Hessian, giving a sandwich variance that is robust to misspecification of the Step-3 model. `"opg"` uses the outer product of the case-wise scores, which relies on the information-matrix equality and is valid only when the Step-3 model is correctly specified. If the Hessian cannot be inverted, `"opg"` is used with a warning. Applies to ML covariate models.

`verbose`
: Logical. Print progress and convergence messages.

Value

A list of class `"tse_control"`.

Examples

```
tse_control()
tse_control(step1.maxit = 10000, n_init = 20)
```

---

tse_covariate *Relate latent classes to covariates (Step 3)*

Description

Estimates a multinomial logistic regression of latent class membership on covariates, $P(X = t \mid Z) \propto \exp(Z\gamma_t)$, correcting for the classification error of the Step-2 class assignments. The measurement model is held fixed, so the covariates cannot change the classes.

Usage

```
tse_covariate(
  object,
  formula,
  method = c("ML", "BCH", "none"),
  se = c("corrected", "robust"),
  ref = 1,
  start = NULL,
```

```
  control = NULL,
  data = NULL
)
```

Arguments

`object`	A classification from `tse_classify()`.
`formula`	One-sided formula for the covariates, e.g. `~ age + sex`. Factors, interactions, and transformations are allowed.
`method`	Step-3 estimator: `"ML"`, `"BCH"`, or `"none"` (see Details).
`se`	Standard errors: `"corrected"` or `"robust"` (see Details).
`ref`	Reference class of the multinomial logit: a class number or label such as `"C2"`.
`start`	Optional starting values: a (Q+1) x (T-1) coefficient matrix (Q covariates plus the intercept, T classes). By default, the two-step estimates (Bakk and Kuha 2018) are used.
`control`	Estimation settings; default: those of the measurement model. See `tse_control()`.
`data`	Optional data frame with the covariates: the classified data (the same rows, in the same order) with any additional columns. Omitted: the data stored in `object`.

Details

Estimators (`method`):

- `"ML"` (default): the bias-adjusted maximum likelihood estimator of Vermunt (2010), which treats the assigned class as an indicator of the true class with known classification-error probabilities.
- `"BCH"`: the Bolck-Croon-Hagenaars estimator (Bolck, Croon, and Hagenaars 2004; Vermunt 2010), which reweights the assignments with the inverse of the classification-error matrix.
- `"none"`: the uncorrected three-step estimator (a weighted multinomial logit of the assigned classes), which is biased toward zero; provided for comparison.

Standard errors (`se`): `"corrected"` (default) adds the uncertainty of the Step-1 measurement model (Bakk, Oberski, and Vermunt 2014) to the robust (sandwich) Step-3 variance; `"robust"` omits it. For `"BCH"` the robust variance is used, which accounts for the Step-1 uncertainty through the weights (Vermunt 2010); for `"none"` the robust variance is used.

Value

A `tseLCA_covariate` object; see `coef.tseLCA_structural()`, `summary.tseLCA_structural()`, `predict.tseLCA_covariate()`, and `anova.tseLCA_covariate()`. Pass it to `tse_distal()` to also model a distal outcome.

References

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., Oberski, D. L., & Vermunt, J. K. (2014). Relating latent class assignments to external variables: Standard errors for correct inference. *Political Analysis*, 22(4), 520–540. doi:10.1093/pan/mpu003

Bolck, A., Croon, M., & Hagenaars, J. (2004). Estimating latent structure models with categorical variables: One-step versus three-step estimators. *Political Analysis*, 12(1), 3–27. doi:10.1093/pan/mph001

Vermunt, J. K. (2010). Latent class modeling with covariates: Two improved three-step approaches. *Political Analysis*, 18(4), 450–469. doi:10.1093/pan/mpq025

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
cl <- tse_classify(m)
fit <- tse_covariate(cl, ~ Zp)
summary(fit)
confint(fit)
predict(fit, newdata = data.frame(Zp = 1:5))

# BCH, and the uncorrected estimator for comparison
coef(tse_covariate(cl, ~ Zp, method = "BCH"))
coef(tse_covariate(cl, ~ Zp, method = "none"))
```

---

tse_distal *Relate latent classes to a distal outcome (Step 3)*

Description

Estimates the class-specific distribution of a distal outcome, correcting for the classification error of the Step-2 class assignments (Bakk, Tekle, and Vermunt 2013). Given a covariate model from `tse_covariate()`, the class prior depends on the covariates and the covariate-model uncertainty is propagated to the distal estimates. The Step-2 assignments and classification errors are those of the classification (they use the indicators only); only the class prior is covariate-adjusted.

Usage

```
tse_distal(
  object,
  formula,
  family = "gaussian",
  method = NULL,
  se = NULL,
  control = NULL,
  data = NULL
)
```

Arguments

object	A classification from tse_classify(), or a covariate model from tse_covariate() (combined model).
formula	Zo ~ 1, with the distal outcome on the left-hand side.
family	Distribution of the outcome within classes: "gaussian" (default), "poisson", "binomial", "multinomial" (nominal outcome), or the corresponding family object (gaussian(), poisson(), binomial(); canonical links only).
method, se	As for tse_covariate(). For a combined model they default to those of the covariate model.
control	Estimation settings; default: those of object.
data	Optional data frame with the distal outcome, as in tse_covariate().

Details

The class-specific parameters are means (gaussian, with a common within-class variance, reported as $sigma2), log means (poisson), logits (binomial), or category probabilities ("multinomial"). The estimators and standard errors are as for tse_covariate(). Use omnibus_test() to test whether the outcome differs across classes.

Value

A tseLCA_distal object, or a tseLCA_both object when object is a covariate model.

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

Examples

```
d <- generate_data(500, "high", "distal", seed = 2)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
fd <- tse_distal(tse_classify(m, assignment = "proportional"), Zo ~ 1)
summary(fd)
omnibus_test(fd)
```

---

tse_lca *Fit a latent class measurement model (Step 1)*

Description

Estimates the measurement model: the class sizes and the class-conditional item-response probabilities of the indicators on the left-hand side of formula. This is the first step of three-step estimation; covariates and distal outcomes are related to the classes afterwards, holding this model fixed.

Usage

```
tse_lca(
  formula,
  data,
  nclass,
  start = NULL,
  missing = c("listwise", "fiml"),
  control = tse_control()
)
```

Arguments

formula	A formula `cbind(Y1, Y2, ...) ~ 1` naming the indicators. The measurement model has no covariates.
data	A data frame.
nclass	Number of latent classes, or a vector of numbers of classes to compare (e.g. `1:6`).
start	Optional fixed starting point for the EM algorithm (single `nclass` only): an integer vector with one class per row of `data`, or a matrix of item-response probabilities P(Y = k | X = t) with one row per item category (in the order of the indicators) and one column per class, such as `item_probs()` of a fitted model (whose binary items have one row, P(Y = 1 | X = t)). Bypasses the default k-means initialization.
missing	Handling of missing indicator values: `"listwise"` (default) drops rows with any missing indicator; `"fiml"` keeps rows with at least one observed indicator (full-information maximum likelihood).
control	Estimation settings, see `tse_control()`.

Details

With a vector `nclass`, a model is fitted for each number of classes and the result is a class-enumeration table of fit statistics (see Details).

The number of classes is chosen from the measurement model alone, before any structural variables are considered, typically by the BIC, the interpretability of the classes, and their separation; see Nylund, Asparouhov, and Muthén (2007) and Masyn (2013). The enumeration table reports, for each number of classes, the log-likelihood, number of free parameters, AIC, BIC, sample-size adjusted BIC (SABIC; Sclove 1987), entropy R^2, and the smallest estimated class proportion. The one-class model is the independence model, fitted in closed form.

Indicators may be factors, logicals, character, or numeric codes; their categories are stored with the model and reused when the model is applied to new data (see `predict.tseLCA_measurement()`) or in later steps.

Value

For a single `nclass`, a `tseLCA_measurement` object (see `class_sizes()`, `item_probs()`, `posterior()`, `predict()`; it keeps `data` for `tse_classify()`); for several, a `tseLCA_select` object: the enumeration table with the fitted models, see `best_model()`.

References

Masyn, K. E. (2013). Latent class analysis and finite mixture modeling. In T. D. Little (Ed.), *The Oxford Handbook of Quantitative Methods*, Vol. 2, 551–611. Oxford University Press.

Nylund, K. L., Asparouhov, T., & Muthén, B. O. (2007). Deciding on the number of classes in latent class analysis and growth mixture modeling: A Monte Carlo simulation study. *Structural Equation Modeling*, 14(4), 535–569. doi:10.1080/10705510701575396

Sclove, S. L. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika*, 52(3), 333–343. doi:10.1007/BF02294360

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)

# Class enumeration
sel <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 1:4)
sel
plot(sel)

# The selected model
m <- best_model(sel, criterion = "BIC")
m
class_sizes(m)
item_probs(m)
head(predict(m, newdata = d[1:5, ]))
```

---

tse_twostep *Two-step estimates of covariate effects*

Description

Estimates the multinomial logit of class membership on covariates with the measurement-model parameters held fixed at their Step-1 values (Bakk and Kuha 2018). Unlike the three-step estimators, the indicators enter the Step-2 likelihood directly, so no classification step is needed.

Usage

```
tse_twostep(object, formula, ref = 1, se = FALSE, control = NULL)
```

Arguments

`object`	A measurement model from `tse_lca()` (it must keep its data).
`formula`	One-sided covariate formula.
`ref`	Reference class of the multinomial logit.
`se`	Logical. If `TRUE`, the estimates and their standard errors (corrected for the Step-1 uncertainty) are obtained with the two-step estimator of **multilevLCA**, initialized at this model's classes; its measurement model is checked against `object`. If `FALSE` (default), only the estimates are computed, and the variance is `NA`.
`control`	Estimation settings; default: those of `object`.

Value

A `tseLCA_twostep` object (also a `tseLCA_covariate`).

References

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

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
coef(tse_twostep(m, ~ Zp))
```

---

tseLCA *Three-step latent class analysis in one call*

Description

Fits the measurement model, classifies the observations, and relates the classes to covariates and/or a distal outcome, all from one formula. This is a convenience wrapper around the step-wise functions `tse_lca()`, `tse_classify()`, `tse_covariate()`, and `tse_distal()`, which remain available for inspecting each step (see `measurement()`).

Usage

```
tseLCA(
  formula,
  data,
  nclass,
  family = "gaussian",
  method = c("ML", "BCH", "none"),
  se = c("corrected", "robust"),
  assignment = c("modal", "proportional"),
  ref = 1,
  missing = c("listwise", "fiml"),
  start = NULL,
  control = tse_control()
)
```

Arguments

`formula`	`cbind(indicators) ~ covariates | distal outcome`; see Details.
`data`	A data frame.
`nclass`	Number of latent classes.
`family`	Distribution of the distal outcome; see `tse_distal()`.

`method`	Step-3 estimator: `"ML"`, `"BCH"`, or `"none"`; see `tse_covariate()`.
`se`	Standard errors: `"corrected"` or `"robust"`.
`assignment`	Step-2 class assignment: `"modal"` or `"proportional"`.
`ref`	Reference class of the covariate model.
`missing`	Missing indicator values: `"listwise"` or `"fiml"`; see `tse_lca()`.
`start`	Optional Step-1 starting values; see `tse_lca()`.
`control`	Estimation settings, see `tse_control()`.

Details

The formula has up to three parts: `indicators ~ covariates | outcome`.

- Left-hand side: the indicators, `cbind(Y1, Y2, ...)`.
- First right-hand side part: the covariates of class membership, with the usual formula syntax (`1` for none).
- Optional second part: the name of a distal outcome.

For example, `cbind(Y1, Y2, Y3) ~ age + sex | income` relates the classes to the covariates `age` and `sex` and to the distal outcome `income`; `cbind(Y1, Y2, Y3) ~ 1 | income` has only the distal outcome; and `cbind(Y1, Y2, Y3) ~ 1` fits the measurement model alone.

The number of classes is chosen beforehand from the measurement model, for example with `tse_lca(..., nclass = 1:6)`.

Value

A `tseLCA_measurement`, `tseLCA_covariate`, `tseLCA_distal`, or `tseLCA_both` object, depending on the formula. Its components are available with `measurement()`, `classification()`, `covariate()`, and `distal()`.

Examples

```
d <- generate_data(500, "high", "covariate", seed = 1)
d$Zo <- draw_Zo(d$X, bk2018_params$distal_params)

fit <- tseLCA(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ Zp | Zo, data = d, nclass = 3)
summary(fit)
measurement(fit)
classification(fit)

# the same model, step by step
m <- tse_lca(cbind(Y1, Y2, Y3, Y4, Y5, Y6) ~ 1, data = d, nclass = 3)
fc <- tse_covariate(tse_classify(m), ~ Zp)
fb <- tse_distal(fc, Zo ~ 1)
```

`vcov.tseLCA_structural` *Variance-covariance matrix of a fitted tseLCA model*

Description

Row and column names match `coef()`.

Usage

```
## S3 method for class 'tseLCA_structural'
vcov(
  object,
  component = c("all", "covariate", "distal"),
  step = c("three_step", "two_step"),
  ...
)

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

Arguments

`object`	A fitted `tseLCA` object.
`component`	For `tseLCA_both` objects: `"all"` (default; covariate then distal coefficients), `"covariate"`, or `"distal"`.
`step`	`"three_step"` (default) or `"two_step"` (the two-step estimates used to initialize Step 3; covariate models only).
`...`	Further arguments (currently unused).
`boundary.tol`	Measurement models only: parameters within this tolerance of 0 or 1 are treated as fixed. Default `1e-2`.

Details

- Measurement models: the BHHH variance matrix of the log-ratio parameters (attribute `"parameterization"` records the scale).
- Covariate and distal-outcome models: the Step-3 variance matrix, which includes the correction for Step-1 uncertainty unless the model was fitted with `use.simple.cov = TRUE`. For `family = "multinomial"` it is on the probability scale and rank-deficient (each class's probabilities sum to one).
- `tseLCA_both` with `component = "all"`: the covariate and distal blocks on the diagonal; the cross-covariances between the two sets of parameters are not computed and are `NA`.

Value

A named square matrix.

Examples

```
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)
```

Index

[[.tseLCA_select *(best_model)*, 5

anova.tseLCA_covariate, 3
anova.tseLCA_covariate(), *41*
as.data.frame.tseLCA_select
(best_model), 5
as_tse_lca, 4

best_model, 5
best_model(), *44*
bk2018_params, 6, *17, 18*

class_sizes, 7
class_sizes(), *9*, *44*
classes *(posterior)*, 26
classification *(measurement)*, 23
classification(), *47*
coef(), *48*
coef.summary.tseLCA_structural
(summary.tseLCA_structural), 30
coef.tseLCA_measurement
(coef.tseLCA_structural), 8
coef.tseLCA_structural, 8
coef.tseLCA_structural(), *41*
covariate *(measurement)*, 23
covariate(), *47*

distal *(measurement)*, 23
distal(), *47*
draw_classes, 10
draw_classes_given_Zp, 10
draw_indicators, 11
draw_Zo, 11
draw_Zp, 12

fitted.tseLCA_measurement
(predict.tseLCA_measurement),
28
fitZ_from_fit0, 12
fitZ_from_fit0(), *20*, *36*
fitZ_from_multiLCA, 14
fitZ_from_multiLCA(), *33*, *36*

generate_all_conditions, 16
generate_data, 17
generate_data(), *6*, *17*
graphics::matplot(), *6*

item_probs *(class_sizes)*, 7
item_probs(), *4*, *9*, *44*

lca_step1, 19
lca_step1(), *15*, *32*, *34*, *36*
lca_step1_startval(), *15*, *20*, *32*
logLik.tseLCA, 21

make_rho, 22
measurement, 23
measurement(), *46*, *47*
mnl_probs, 24
multilevLCA::multiLCA(), *16*, *20*

nobs.tseLCA *(logLik.tseLCA)*, 21

omnibus_test, 25, *34*
omnibus_test(), *43*

plot.tseLCA, 26
plot.tseLCA_select *(best_model)*, 5
posterior, 26
posterior(), *44*
predict(), *44*
predict.tseLCA_covariate, 27
predict.tseLCA_covariate(), *41*
predict.tseLCA_measurement, 28
predict.tseLCA_measurement(), *44*
print.summary.tseLCA_measurement
(summary.tseLCA_structural), 30
print.summary.tseLCA_structural
(summary.tseLCA_structural), 30
print.tseLCA_classify *(tse_classify)*, 38

print.tseLCA_measurement
(summary.tseLCA_structural), 30
print.tseLCA_select (best_model), 5
print.tseLCA_structural
(summary.tseLCA_structural), 30

relevel.tseLCA_covariate, 29

stats::AIC(), *21*
stats::BIC(), *21*
stats::confint(), *8*
stats::printCoefmat(), *30*
summary.tseLCA_measurement
(summary.tseLCA_structural), 30
summary.tseLCA_structural, 30
summary.tseLCA_structural(), *41*

three_step, 31
three_step(), *14, 15*
tse_classify, 38
tse_classify(), *5*, *41*, *43, 44*, *46*
tse_control, 39
tse_control(), *36*, *38*, *41*, *44*, *47*
tse_covariate, 40
tse_covariate(), *3*, *28, 29*, *42, 43*, *46, 47*
tse_distal, 42
tse_distal(), *41*, *46*
tse_lca, 43
tse_lca(), *4, 5*, *20*, *32*, *38*, *45–47*
tse_twostep, 45
tse_twostep(), *14*, *16*
tseLCA, 46
tseLCA(), *23*, *35*

vcov(), *4, 5*, *8*
vcov.tseLCA_measurement
(vcov.tseLCA_structural), 48
vcov.tseLCA_structural, 48