Package 'latents'

November 2, 2026

Title Multilevel Latent Profile, Class and Transition Models

Version 1.1.9

Description Fits single-level and multilevel latent profile and class models, latent transition models, and mixture regression and growth models for studying unobserved heterogeneity. Supported measurement models accommodate continuous, categorical, ordinal, count and mixed indicators. Provides model comparison, parameter inference, classification diagnostics and graphical summaries, with additional methods for membership covariates, staged estimation and analyses of external variables that account for classification error. Observed-data likelihood methods are available for incomplete indicators, and supported transition models can be represented as networks using the 'tna' package. The multilevel latent class formulation follows Vermunt (2003) <doi:10.1111/j.0081-1750.2003.t01-1-00131.x>.

URL https://pak.dynasite.org/latents/, https://github.com/mohsaqr/latents

BugReports https://github.com/mohsaqr/latents/issues

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, grDevices, utils, parallel

Suggests testthat (>= 3.0.0), knitr, rmarkdown, tna, mclust, mice, ggplot2, MASS, withr, numDeriv

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

LazyData true

NeedsCompilation no

Author Mohammed Saqr [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-5881-3109>), Sonsoles López-Pernas [aut] (ORCID: <https://orcid.org/0000-0002-9621-1392>)

Maintainer Mohammed Saqr <saqr@saqr.me>

Repository CRAN

Date/Publication 2026-11-01 23:30:02 UTC

Contents

latents-package . . . 3
bootstrap_lrt . . . 4
compare_models . . . 8
course_engagement . . . 11
descriptives . . . 13
diagnostics . . . 15
enumerate_classes . . . 18
enumerate_lca . . . 22
enumerate_lpa . . . 23
enumerate_regressions . . . 24
fit_staged . . . 27
get_candidate . . . 31
get_group_tna . . . 32
get_results . . . 33
get_tna . . . 45
growth_schools . . . 47
growth_scores . . . 48
latents-conditions . . . 49
latents-plots . . . 57
lca . . . 66
lpa . . . 67
lta . . . 68
mixture_regression . . . 75
multilca . . . 81
multilpa . . . 83
parameter_inference . . . 96
pool_imputations . . . 102
predict.latents_mixture_regression . . . 104
prior_control . . . 107
r3step . . . 109
report . . . 113
sensitivity . . . 114
srl . . . 117
starting_values . . . 118
student_esm . . . 119
study_hours . . . 121
three_step . . . 122
vcov.multilpa_covariates . . . 126

Index **129**

`latents-package` *Latent Profile, Class and Transition Models*

Description

latents is an R package for modelling unobserved heterogeneity in response patterns, regression relationships and trajectories. It supports latent profile and class analysis, multilevel models, latent transition analysis, and mixture regression and growth models.

The package combines model estimation and comparison with tools for assessing parameter uncertainty, classification quality and solution stability. It also supports analyses relating latent membership to external variables and provides tables and plots for interpreting and reporting results.

Details

Latent profiles describe heterogeneity in continuous indicators through component-specific distributions. Latent classes describe heterogeneity in discrete response patterns. In the two-level model, observation-level profiles or classes are nested within groups, and latent group classes differ in their conditional probabilities of observation-level membership. The standard two-level formulation shares measurement parameters across group classes. Transition models additionally estimate dependence between latent states at successive occasions.

The number of latent classes and the measurement specification are supplied by the analyst. Estimation uses multiple starting values because different starts can converge to different local maxima. Convergence, replication of the selected likelihood, class sizes, parameter boundaries, and substantive interpretability should be considered together when evaluating a solution. Classification summaries describe uncertainty under the fitted model; they do not establish the substantive validity of the classes.

Model fitting

Use `lpa` or `lca` for single-level models and `multilpa` or `multilca` for two-level models. `lta` fits latent transition models, and `mixture_regression` fits finite mixtures of regression models. The `family` argument of `multilpa` provides additional model families documented on that function's help page. `fit_staged` estimates a two-level model with measurement parameters fixed at their first-stage estimates.

Model evaluation and inference

`enumerate_classes`, `enumerate_lpa`, `enumerate_lca`, and `enumerate_regressions` fit candidate model sets. `compare_models` summarizes selected models, and `bootstrap_lrt` provides parametric bootstrap likelihood ratio tests for supported comparisons.

`get_results` extracts model-specific tables; `diagnostics` and `sensitivity` assess fitted solutions and sensitivity to starting values. `parameter_inference` provides parameter estimates, standard errors, and intervals for supported model families. `three_step` and `r3step` relate external variables to latent membership with correction for classification error. `pool_imputations` combines compatible analyses across imputed datasets.

Interpretation and assumptions

The measurement distribution, conditional independence assumptions, covariance constraints, and treatment of time depend on the selected model. Consult the fitting function for these assumptions and for restrictions on missing data, sampling weights, covariates, and parameter inference. Posterior class assignments are estimates with uncertainty. Class labels are arbitrary identifiers and should be interpreted through the estimated response patterns. Numerical fit and classification separation alone do not establish causal effects or distinct populations.

Getting started

The vignettes `vignette("lpa", package = "latents")` and `vignette("lca", package = "latents")` introduce the main workflows; class enumeration and latent transition analysis are covered in articles on the package website, https://pak.dynasite.org/latents/articles/. See `latents-plots` for graphical summaries, `latents-predict` for prediction, and `latents-conditions` for diagnostic conditions.

References

Vermunt, J. K. (2003). Multilevel latent class models. *Sociological Methodology*, 33, 213–239. doi:10.1111/j.00811750.2003.t01100131.x.

`bootstrap_lrt` *Parametric bootstrap likelihood-ratio test for nested mixture models*

Description

The parametric bootstrap likelihood-ratio test evaluates whether a more flexible mixture model provides an improvement in fit that is inconsistent with a simpler, nested model. `bootstrap_lrt()` compares two fitted models using twice the difference between their maximized log likelihoods. In class enumeration, this comparison assesses the evidence for adding a latent profile or group class.

The function generates datasets from the fitted null model and refits both candidate models to each dataset. These simulated comparisons approximate the distribution of the likelihood-ratio statistic under the null hypothesis. The observed statistic is then evaluated against this distribution to obtain a bootstrap p-value. This approach avoids relying on the usual chi-squared approximation, which is generally inappropriate when comparing different numbers of mixture components.

The result includes the observed test statistic, bootstrap p-value, Monte Carlo standard error, and diagnostics for the simulated refits. The function supports profile and class models, latent transition models, and group-class model families, subject to the nesting and compatibility requirements below.

Usage

```
bootstrap_lrt(
  null_model,
  alternative_model,
  data = NULL,
```

```
  iter = 199L,
  n_starts = NULL,
  max_iter = NULL,
  tol = NULL,
  seed = NULL
)
```

Arguments

null_model	A converged fitted model representing the null hypothesis. Supported inputs are profile or class models, with or without membership covariates; transition models from lta(); and group-class models fitted through multilpa(). See *Supported comparisons* for nesting requirements.
alternative_model	A compatible, more flexible model fitted to the same data, within which null_model is nested.
data	The data frame used to fit both models. Required for transition models, whose simulations retain the observed groups, occasions and covariates. For ordinary profile and class models, omission reconstructs the data from the null fit, or from the alternative for membership-covariate models. For group-class models, the stored sufficient statistics are used when data is omitted. Supplied data are checked against the fitted models.
iter	Number of simulated datasets; default 199. Profile, class and group-class comparisons require at least two replicates; transition comparisons accept any positive integer. Larger values reduce Monte Carlo uncertainty in the estimated p-value.
n_starts	Number of starts for each simulated fit. With NULL (the default), each model retains its original number of starts and search strategy. A supplied number applies to both models.
max_iter	Maximum iterations for each simulated fit. With NULL (the default), each model retains its original iteration limit.
tol	Convergence tolerance for each simulated fit. With NULL (the default), each model retains its original tolerance.
seed	Optional integer seed accepted by set.seed(). The caller's random-number state is restored on exit.

Value

An object of class multilpa_bootstrap_lrt containing the observed statistic, bootstrap p-value, Monte Carlo standard error, numbers of valid and invalid replicates, and replicate-level diagnostics. The fixed component identifies any measurement blocks held constant.

as.data.frame() returns a one-row test summary. get_results(result, what = "replicates") returns one row per simulated replicate. summary() combines the test summary and replicate diagnostics; plot() displays the simulated reference distribution and observed statistic.

Interpretation

A small p-value provides evidence against the null model in favour of the specified alternative, subject to the model assumptions and the quality of estimation. It does not establish that the alternative has the correct number of classes or fits the data adequately in an absolute sense. Interpretation should consider convergence, replication across starting values, and the frequency of invalid bootstrap replicates. Increasing the number of replicates reduces simulation uncertainty but does not resolve systematic estimation problems.

Supported comparisons

For ordinary profile and class models, with or without membership covariates, the models must differ by exactly one observation-level profile or class, or by exactly one group class. They must use the same observations, group layout, covariance structure, centering and scaling, missing-data handling, and `drop_empty` setting. Grand-mean centering is repeated in each simulated refit. Person-centred or person-scaled fits are not supported, because the model does not specify a distribution for the group baselines removed by these transformations.

Transition models fitted with `lta()` support comparisons in which the null specification is nested within the alternative. Supported differences include class counts and restrictions on measurement, transition probabilities, Markov order, and covariate effects. Both models must use compatible indicators, grouping, occasion layout, and measurement distributions. The original `data` must be supplied for transition comparisons.

Group-class models fitted through the `family` argument of `multilpa()` support comparisons across class counts and parameter restrictions. The null must be nested within, and distinct from, the alternative: the alternative must have at least as many group classes and cannot impose additional restrictions on the class-specific means or variance components. Models from different families cannot be combined arbitrarily.

For membership-covariate models, covariates are held at their observed values when simulating data. Group classes and profiles are drawn using the fitted membership probabilities. Both models must use the same profile covariates, group covariates and slope specification, except that a one-group-class null may be compared with an alternative that adds group covariates or slopes that vary by group class.

For models fitted with `fixed`, including those from `fit_staged()`, each replicate retains the same fixed blocks at the same values. The resulting p-value is conditional on that measurement solution; uncertainty in its estimation is not propagated. Pairs for which nesting cannot be established are rejected.

Bootstrap calculation

The observed statistic is `2 * (logLik(alternative_model) - logLik(null_model))`. Datasets are simulated under the fitted null model, retaining the group structure and other design features required by the model family. Both models are refitted to each simulated dataset and the statistic is recalculated. By default, each model retains its original search controls; `n_starts`, `max_iter` and `tol` can override the corresponding settings for both models.

The bootstrap p-value is `(1 + b) / (V + 1)`, where `b` is the number of valid simulated statistics at least as large as the observed statistic and `V` is the number of valid replicates. When all replicates are valid, `V = iter`. The reported Monte Carlo standard error summarizes uncertainty from the finite number of simulated replicates.

Estimation failures

Both original models must have converged. A simulated replicate is invalid if a refit fails, does not converge, or produces an alternative-model likelihood below the null-model likelihood beyond numerical tolerance. The result reports `n_valid` and `n_invalid`; `latents_failed_replicates` warns when any replicate is invalid. The p-value uses only valid replicates and is `NA` when fewer than half of the requested replicates are valid.

Excluding invalid replicates conditions the reference distribution on successful estimation and may affect calibration. Report the invalid proportion and examine the replicate diagnostics before interpreting the test.

For membership-covariate models, a simulated refit is considered converged for this test when its likelihood has converged, even if a membership logit is still changing. This can occur with an empty or separated class in an alternative model. The `logits_settled` column records whether the logits also converged. The original covariate models must be fully converged.

Missing-data assumptions

Fits estimated with `missing = "fiml"` are supported when both models use that setting. Each simulated dataset retains the original missing-value pattern before refitting. Calibration requires missingness independent of the indicators and latent classes, conditional on any fixed covariates. A fixed missingness pattern does not reproduce a missing-at-random (MAR) mechanism that depends on observed indicators or a missing-not-at-random (MNAR) mechanism. Valid FIML estimation under MAR does not by itself establish validity of this bootstrap under MAR.

Conditions

For ordinary profile and class comparisons, `latents_bad_nesting` indicates incompatible class counts, fixed measurement blocks, covariate specifications, or parameter counts. `latents_incomparable_models` indicates incompatible data or fitting specifications. Transition and group-class comparisons use their family-specific nesting checks and raise `latents_bad_argument` for unsupported nesting requests. Transition models also raise `latents_bad_argument` when `data` is omitted.

`latents_unsupported_bootstrap` identifies unsupported person centering or scaling. Models with a prior, sampling weights or a noise component raise `latents_unsupported_prior`, `latents_unsupported_weights` or `latents_unsupported_noise`, respectively. `latents_bad_data` identifies unsupported missing indicators, and `latents_bad_inference_data` identifies supplied data that do not reproduce the original fitting data or likelihood.

`latents_no_converge` indicates that an original model has not converged; for ordinary profile and class models it also identifies active variance bounds. `latents_reversed_likelihood` indicates that the alternative has a lower likelihood than the null beyond numerical tolerance. Invalid seeds or incompatible model families raise `latents_bad_argument`. `latents_failed_replicates` warns about invalid simulated replicates; see *Estimation failures* for their treatment.

References

Nylund, K. L., Asparouhov, T., & Muthen, 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.

See Also

enumerate_classes() for fitting candidate models; compare_models() for information-criterion comparisons.

Examples

```
set.seed(1)
example_data <- data.frame(
  school = rep(seq_len(10), each = 10),
  score = rnorm(100, rep(c(-2, 2), each = 50))
)
smaller <- multilpa(example_data, "score", "school", n_profiles = 1,
                    n_group_classes = 1, n_starts = 2, seed = 1)
larger <- multilpa(example_data, "score", "school", n_profiles = 2,
                   n_group_classes = 1, n_starts = 2, seed = 1)
# `iter` is small so the example runs quickly; use many more for inference.
comparison <- bootstrap_lrt(smaller, larger, iter = 9, n_starts = 1,
                            max_iter = 2000, tol = 1e-6, seed = 1)
comparison
```

---

compare_models *Compare the fit and classification of mixture models*

Description

compare_models() compares alternative mixture models fitted to the same observed data. It supports assessment of the number of latent classes and other model specifications by presenting evidence about model fit, complexity, classification, and convergence.

Information criteria summarise the trade-off between fit and the number of estimated parameters. Classification measures provide complementary information about the separation and relative size of the classes, while convergence summaries help identify estimation problems that may compromise the comparison. These measures address different aspects of a solution and should be interpreted together.

The function accepts separately fitted models or candidates from an enumeration grid and returns their comparison statistics in a single table. BIC differences express each model's distance from the smallest BIC among the candidates; normalised BIC weights summarise their relative support within that set. The function does not select a final model.

Usage

```
compare_models(
  ...,
  n_profiles = NULL,
  n_group_classes = NULL,
  cutoff = 0.8,
  test = c("none", "lrt")
)
```

Arguments

`...`	Either an enumeration grid from `enumerate_classes()` followed by the candidates to compare (none for all of them, or a selection through `n_profiles` and `n_group_classes`); or two or more fits from `multilpa()`, `lpa()`, `lca()`, `lta()` or `get_candidate()`; or two or more fits from `mixture_regression()` (with or without `random`). Named fits are labelled by their names, unnamed fits by their expression, and candidates by their candidate name. Labels must be distinct.
`n_profiles, n_group_classes`	With a grid, the class counts of the candidates to compare; `NULL` (the default) takes every count. Candidates whose fit failed are left out, with a message.
`cutoff`	For profile and class models, the average posterior probability below which a class is counted in `avepp_below` and `group_avepp_below`; a single number in (0, 1], default 0.8. Not used for mixture regressions.
`test`	`"none"` (the default) for the comparison table alone, or `"lrt"` to add a likelihood-ratio test of each profile, class or transition model against the one before it (see *Details*).

Details

`delta_bic` is the difference from the smallest BIC among the supplied candidates. `bic_weight` is the Schwarz weight `exp(-delta_bic / 2)`, normalised across those candidates (Wagenmakers and Farrell, 2004). Incompatible data produce an error.

Candidates of an enumeration grid are selected by name, as the grid's `candidate` column shows them (`compare_models(grid, "profiles3_groups4", "profiles3_groups5")`), or by class counts (`compare_models(grid, c(3, 4), c(3, 5))`, where a single number means one group class). Alternatively, `n_profiles` and `n_group_classes` select candidates: `compare_models(grid, n_profiles = 3)` compares every candidate with 3 profiles, and `compare_models(grid)` compares them all. Latent class grids work the same way, with `classes` in the names, and the `n_profiles` column counting classes. Fits from `multilpa()`, `lpa()`, `lca()`, `lta()` or `get_candidate()` can also be passed directly.

With `test = "lrt"`, each model is tested against the model in the row before it, as `stats::anova()` does: `lr_statistic` is twice the difference in their log likelihoods, `lr_df` the difference in their numbers of free parameters, and `lr_p_value` its upper chi-square tail (Wilks, 1938). List the models from the most restricted to the most general. A model is nested in the next when the two have the same indicators, coding, estimation settings and numbers of profiles and group classes, and the first only restricts the second: a covariance structure restricted letter by letter (EEI within VVI or EEE, and both within VVV), covariances of chosen pairs (`local_dependence`) within more pairs or a full covariance, fewer joint item pairs, direct effects or membership covariates, profile slopes shared across group classes, count dispersions shared across profiles; and for `lta()`, measurement invariant within measurement by occasion, homogeneous within occasion-specific transitions, first within second order, and fewer transition or initial covariates. These restrictions hold parameters at interior values, where the chi-square reference applies. A different number of profiles or group classes, or a stayer class, puts the null on the boundary of the parameter space, where it does not: those comparisons are refused and belong to `bootstrap_lrt()` or `get_results(grid, "lmr")`.

Value

A base `data.frame` of class `latents_comparison`, one row per model.

For mixture regressions the columns are `model`, `n_classes`, `random`, `n_parameters`, `log_likelihood`, `aic`, `bic`, `sabic`, `icl`, `entropy`, `smallest_share`, `delta_bic`, `bic_weight` and `converged`.

For profile and class models they are `model`, `n_profiles`, `n_group_classes`, `structure` (the covariance structure, `NA` without continuous indicators), `n_parameters`, `log_likelihood`, every information criterion of `enumerate_classes()`'s candidates table, `profile_entropy`, `group_entropy`, `avepp_min`, `avepp_max`, `avepp_mean`, `avepp_below` and the same four prefixed `group_`, `smallest_share` (the smallest estimated profile share, relative to all observation mass including any noise component), `converged`, `n_best_replicated`, and `delta_bic` and `bic_weight`, computed on `bic_groups` (the BIC that counts groups as the independent units; on a single-level model, the ordinary BIC). The values are those of the grid's candidates table.

With `test = "lrt"`, three more columns: `lr_statistic`, `lr_df` and `lr_p_value`, `NA` in the first row.

Raises `latents_bad_argument` for fits that cannot be compared, for fewer than two models, for duplicate labels, for a group-class family grid (its own table, `as.data.frame(grid)`, compares its candidates), for candidates named together with `n_profiles` or `n_group_classes`, and for `n_profiles` or `n_group_classes` without a grid; `latents_unknown_candidate` for a name, counts or selection not in the grid; `latents_failed_candidate` for a named candidate that could not be fitted; `latents_incomparable_models` for fits to different data. A `cutoff` outside (0, 1] fails the argument checks.

With `test = "lrt"`: `latents_bad_nesting` for consecutive models that are not nested as described in *Details*, or that differ in their numbers of profiles or group classes; `latents_reversed_likelihood` when the more general model has the lower likelihood beyond the fits' tolerance; `latents_no_converge` for a fit that did not converge; `latents_unsupported_weights`, `latents_unsupported_prior` and `latents_unsupported_noise` for fits with sampling weights, a prior or a noise component; `latents_bad_argument` for mixture regressions. A fit with a variance at `min_variance` warns `latents_boundary`.

Interpretation

Models must use the same outcomes or indicators and observational units, and their likelihoods must be comparable. Lower information criteria favour a model relative to the alternatives considered, but do not establish adequate fit or substantive validity. Classification separation should inform interpretation rather than serve as the sole basis for choosing the number of classes. Final selection also requires examination of estimation stability, model assumptions, and the meaning of the resulting classes.

References

Wilks, S. S. (1938). The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1), 60–62.

Wagenmakers, E.-J., & Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, 11, 192–196.

Examples

```
# Compare trajectory and random-intercept growth models for 60 students.
few_students <- subset(growth_scores, student <= 60)
trajectories <- mixture_regression(score ~ wave, few_students, n_classes = 2,
                                   id = "student", class_level = "group",
                                   seed = 1)
growth <- mixture_regression(score ~ wave, few_students, n_classes = 2,
                             id = "student", class_level = "group",
                             random = "intercept", n_starts = 3, seed = 1)
compare_models(trajectories = trajectories, growth = growth)

# Candidates of an enumeration grid, by name, by class counts, or selected
grid <- enumerate_classes(subset(course_engagement, student <= 30),
                          c("browse", "lectures"), "student",
                          n_profiles = 2:3, n_group_classes = 1:2,
                          model = "VVI", n_starts = 2, seed = 1)
compare_models(grid, "profiles2_groups2", "profiles3_groups2")
compare_models(grid, c(2, 2), c(3, 2))
compare_models(grid, n_group_classes = 2)

# Is a covariance structure that varies by profile needed? A
# likelihood-ratio test of each structure against the one before it.
shared <- lpa(faithful, c("eruptions", "waiting"), n_profiles = 2,
              model = "EEI", n_starts = 2, seed = 1)
varying <- lpa(faithful, c("eruptions", "waiting"), n_profiles = 2,
               model = "VVI", n_starts = 2, seed = 1)
compare_models(shared = shared, varying = varying, test = "lrt")
```

---

course_engagement *Student engagement across a sequence of courses*

Description

course_engagement is a simulated dataset for illustrating student engagement across successive courses. Each row represents one student's enrolment in one course, with courses ordered within students. The data include five activity indicators, standardised within course, and a simulated previous-course grade.

The data were generated from two engagement profiles and two student-level classes that differ in the prevalence of those profiles. Engagement can change across a student's course sequence. The generating profile and student-class labels are retained, allowing fitted classifications to be compared with the structure used to generate the data.

The dataset supports examples of multilevel profile analysis, latent transition analysis and analyses relating previous grades to profile membership.

Usage

```
course_engagement
```

Format

A data frame with 1422 rows (106 students, 32 courses, up to 15 enrolments each) and 11 columns:

student Integer student identifier, 1 to 106. The nesting unit; pass it as `id`.

course Integer course identifier, 1 to 32.

sequence Integer position of the course in the student's own order, 1 to 15. Pass it as `time`. Students who took fewer courses have shorter sequences, so the panel is unbalanced.

browse Numeric, simulated browsing activity, standardized within course.

lectures Numeric, simulated lecture-viewing activity, standardized within course.

forum_read Numeric, simulated forum-reading activity, standardized within course.

forum_post Numeric, simulated forum-posting activity, standardized within course.

attendance Numeric, simulated attendance activity, standardized within course. Generated from the four other activity indicators with added noise, introducing within-profile dependence; see *Local dependence*.

previous_grade Numeric, simulated grade in the previous course, standardized across the cohort; an entry grade for the first course. Can be used as a membership predictor through the `profile_covariates` argument of `multilpa()` or in `r3step()`.

engagement Factor, `"disengaged"` or `"engaged"`: the pattern used to generate the observation; retained for assessing class recovery.

student_type Factor, `"committed"` or `"wavering"`: the student's generating class; retained for assessing group-class recovery.

Details

Students take 10 to 15 courses, ordered by `sequence`. Use `id = "student"` for the nesting structure, `time = "sequence"` for occasion order, and `profile_covariates = "previous_grade"` for a membership-covariate model.

In the generating model, students in the `"committed"` class have a stationary probability of an engaged enrolment of approximately 0.8, compared with approximately 0.25 in the `"wavering"` class.

The activity indicators were simulated as continuous values intended to represent log-transformed activity measures, then standardized within course and rounded. Raw event counts were not generated. Each indicator has approximately mean zero and standard deviation one within course; for example, `browse = 1.2` represents a value 1.2 standard deviations above the course mean. The simulation does not include between-course differences in activity volume.

Local dependence

`attendance` is generated from the four click measures, introducing within-profile dependence among the indicators. A model with diagonal residual covariance therefore does not reproduce the generating covariance structure. Use `get_results(fit, "residuals")` to examine residual associations and `covariance_model = "full"` to estimate within-profile covariances. Class enumeration under a diagonal model may represent residual association by additional classes. The two generating profiles therefore do not imply that every fitted specification will favour two profiles.

Source

Simulated by `data-raw/course-engagement.R` in the source repository, seed 2026. The simulation is calibrated to aggregate constants of a real learning-analytics export (scales, spreads, correlations and the persistence of engagement across courses). No row, identifier or value of any real student is included, and the source data are not distributed.

See Also

`vignette("lpa", package = "latents")` for the analysis these data are used in, `multilpa()` for the two-level model, `lta()` for the sequence, and `r3step()` for membership-covariate analysis.

Examples

```
fit <- multilpa(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student", n_profiles = 2, n_group_classes = 2, n_starts = 4,
  seed = 1
)
as.data.frame(fit)
get_results(fit, what = "profile_probabilities")
# Compare fitted assignments with the generating engagement profiles.
get_results(fit, what = "assignments", data = course_engagement,
         truth = "engagement")
```

`descriptives` *Descriptive statistics and intraclass correlations of model variables*

Description

`descriptives()` summarizes the characteristics of variables used in latent profile and class analyses. It supports inspection of variability, observed ranges, and missingness before model estimation, as well as descriptive comparisons across groups.

For numeric variables, the function reports measures of central tendency and dispersion. When a grouping identifier is supplied, it also estimates intraclass correlations to describe clustering within groups. Summaries can be calculated for the full dataset or separately by an observed grouping variable. For supported fitted models, variables can also be summarized by assigned profile or group class.

Usage

```
descriptives(x, ...)

## S3 method for class 'data.frame'
descriptives(x, vars = NULL, id = NULL, by = NULL, ...)
```

Arguments

x	A data frame, or a profile, class or transition fit (from lpa(), lca(), multilpa(), multilca(), or lta() with time-homogeneous transitions), whose own vars and id are then used. A single-level fit has no groups, so its table has no n_groups or icc. Anything else, including a fit of another family, raises latents_not_a_fit; describe the data frame it was fitted to instead.
...	Not used. For a data frame, any further argument raises a latents_unused_argument warning; for a fit, it raises latents_bad_argument.
vars	Character vector of column names to describe; repeated names are described once. Columns must be vectors: matrix-valued columns raise latents_bad_data. For a data frame, defaults to every numeric column other than id and by.
id	Optional. The column identifying the groups the rows are nested in. Supplying it adds n_groups and icc.
by	Optional. On a fitted model, "profile" or "group_class"; on a data frame, a column name. Each variable is then described once per level of it. Rows whose stratifier is NA form their own stratum, labelled NA and reported last, so every input row is counted in exactly one stratum.

Details

Summaries by fitted class use assigned membership and do not incorporate classification uncertainty. They describe the observations assigned to each class and should be distinguished from the model's estimated class-specific parameters. Rows excluded from classification are retained in a separate stratum with a missing class label, following the fit's drop_empty setting.

An intraclass correlation (ICC) near zero indicates little between-group variation in that variable's mean. It does not rule out group classes that differ in other variables, in profile prevalence, or in other features of the distributions.

Value

A base data.frame, one row per variable (or per variable and by level), with the columns

variable character: the column described.

n integer: observations with a value.

n_missing integer: observations without one.

mean,sd,min,max numeric; NA_real_ for a variable that is not numeric.

n_distinct integer: distinct observed values.

n_groups,icc present only when id is known. icc is the one-way random-effects estimate. It can be negative when the between-group mean square is smaller than the within-group mean square; this does not represent a negative population variance. Rows with a missing group identifier contribute to neither statistic.

With by, a first column named after the stratifier holds each stratum's value (NA for the stratum of missing values). For every variable, n + n_missing summed over the strata equals the number of input rows. When describing a fit by "profile" or "group_class", the model's assignment is used as the stratifier without replacing an indicator of the same name. A by named like one of the summary's own columns raises latents_bad_data. Describing no variables raises latents_nothing_to_describe.

References

Bliese, P. D. (2000). Within-group agreement, non-independence, and reliability. In K. J. Klein & S. W. J. Kozlowski (Eds.), *Multilevel theory, research, and methods in organizations*.

See Also

`multilpa()` for model estimation and `diagnostics()` for diagnostics of a fitted model.

Examples

```
descriptives(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student"
)

# Missing stratification values form a separate stratum.
descriptives(data.frame(cohort = c("A", "A", "B", NA), score = 1:4),
             vars = "score", by = "cohort")

fit <- multilpa(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student", n_profiles = 2, n_group_classes = 2, n_starts = 4,
  seed = 1
)
descriptives(fit)
descriptives(fit, by = "profile")
```

`diagnostics` *Diagnostics of a fitted model*

Description

`diagnostics()` examines the estimation and interpretation of a fitted mixture model. It assesses convergence and replication across starting values, describes the size and separation of the latent classes, and, where supported, examines associations between indicators that are not adequately represented by the model.

Convergence and likelihood replication indicate whether different starting values lead to the same numerical solution. Class sizes and posterior membership probabilities describe how observations are distributed across classes and how uncertain their classification remains. Bivariate residuals compare observed indicator associations with those implied by the fitted model, helping identify departures from the measurement specification.

The function brings these results together in diagnostic tables and, where available, classification plots. For multilevel models, classification summaries distinguish observation-level profiles from group-level classes. Conditions such as incomplete convergence, estimates at parameter bounds, and classes with no assigned observations are identified for further examination.

Usage

```
diagnostics(x, data = NULL, plots = TRUE, by = c("profile", "overall"), ...)
```

Arguments

x	A fitted model of this package.
data	Optional data frame used to fit the model. When omitted, the required columns are reconstructed from the fitted object.
plots	TRUE (the default) draws classification plots for model families that support them: effective profile sizes, posterior probabilities of assigned profiles, case entropy, and the average posterior probability matrix. If ggplot2 is unavailable or the model family has no classification plots, a message is issued and the tables are returned without plots. FALSE returns only the tables.
by	"profile" (the default) assesses each profile's bivariate residuals separately; "overall" pools them. This argument is forwarded to the residual calculations. The classification levels are determined by the fitted model.
...	Not used. For diagnostics() and plot(), any further argument raises latents_bad_argument; style the returned plots with ggplot2.

Details

- **Estimation**: convergence of the selected solution; replication of the best log likelihood across completed starts (Masyn, 2013); failed or unfinished starts; and estimates at parameter bounds. Preliminary screening starts are reported separately from full optimisation runs.
- **Class size**, at each level the fit classifies: the smallest class, the largest gap between modal and model-estimated class proportions, and classes to which no unit is assigned.
- **Classification**: relative entropy (Ramaswamy et al., 1993), which describes classification precision and is not a criterion for the number of classes (Clark & Muthen, 2009); the lowest average posterior probability of the assigned class and the lowest odds of correct classification (Nagin, 2005).
- **Bivariate residuals**: discrepancies between observed and model-implied indicator associations, reported under the "Local independence" domain. Where tests are available, pairs are flagged at 0.05 after Benjamini-Hochberg adjustment (Vermunt & Magidson, 2005). Residuals without an available test are reported descriptively.
- **Information**: log likelihood, number of parameters and BIC, for comparison with other fitted models using compare_models().

The status labels are "fail" for a non-converged selected solution or a class with effective membership below one; "check" for conditions requiring review, including failed or unfinished starts, active parameter bounds, classes with no modal assignments or residual indicator dependence; "ok" when the checked condition is absent; "info" for a descriptive quantity; and "n/a" for an unavailable or inapplicable quantity. An "ok" status applies only to the specified check and does not establish overall model adequacy. Classification summaries use the fitted posterior probabilities, eligible observations and sampling weights. The available diagnostics depend on the model family.

Entropy, classification and average-posterior tables are supplied at the classification levels supported by each family, together with a summary of replication across starts ("replication"). Group-classified regressions report the group level; hierarchical regression and growth models report both

levels. All-empty transition rows follow the fitted drop_empty policy, and zero-weight units are never classified. Bivariate residuals are gathered for families that support them.

Value

An object of class multilpa_diagnostics. The checks table, accessed with get_results(result, "checks"), contains one row per check and columns domain, check, level, value (numeric), display (formatted value), criterion, status, detail and reference. Tables are accessed with get_results(), using what = "checks", "entropy", "classification", "average_posteriors", "residuals", "replication" or "all". A model family without bivariate residuals leaves that table out of "all", and asking for it by name raises latents_no_group_classes; a fit with a noise component has none either, and raises latents_unsupported_noise.

print() groups checks by domain, highlights those requiring review and returns the object invisibly. For supported model families, plot() returns four classification plots as a latents_plots list of ggplot objects; printing the list draws all four plots. as.data.frame() returns the entropy table. summary() returns the diagnostics object.

Interpretation

Numerical convergence, classification separation, and measurement fit address different aspects of a model. A converged solution may contain poorly separated classes, while sharply separated classes may arise under an inadequate measurement specification. The diagnostics should therefore be interpreted jointly, with attention to the study design and the substantive meaning of the estimated classes.

References

Clark, S. L., & Muthen, B. (2009). Relating latent class analysis results to variables not included in the analysis. Unpublished manuscript.

Masyn, K. E. (2013). Latent class analysis and finite mixture modeling. In T. D. Little (Ed.), *The Oxford handbook of quantitative methods* (Vol. 2, pp. 551-611). Oxford University Press.

Nagin, D. S. (2005). *Group-based modeling of development*. Harvard University Press.

Ramaswamy, V., DeSarbo, W. S., Reibstein, D. J., & Robinson, W. T. (1993). An empirical pooling approach for estimating marketing mix elasticities with PIMS data. *Marketing Science*, 12(1), 103-124. doi:10.1287/mksc.12.1.103

Vermunt, J. K., & Magidson, J. (2005). *Latent GOLD 4.0 user's guide*. Statistical Innovations.

See Also

get_results() for these and the other tables, descriptives() for the variables before fitting, and summary() for the estimates.

Examples

```
fit <- multilpa(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student", n_profiles = 2, n_group_classes = 2, n_starts = 4,
```

```
  seed = 1
)
quality <- diagnostics(fit, plots = FALSE)
quality
get_results(quality, "checks")
get_results(quality, what = "classification")
get_results(diagnostics(fit, by = "overall", plots = FALSE), what = "residuals")
```

`enumerate_classes` *Class enumeration and comparison of mixture models*

Description

Class enumeration evaluates alternative numbers of latent classes to inform model selection. `enumerate_classes()` fits a specified set of candidate models and summarizes their fit, classification quality, and estimation diagnostics. It supports latent profile, latent class, and latent transition models, including multilevel specifications.

For multilevel profile and class models, candidates can differ in the number of observation-level profiles or classes, the number of group-level classes, and, for continuous indicators, the within-profile covariance structure. Comparing these specifications allows the analyst to assess both the latent class structure and the assumptions used to represent variation within classes.

The function retains the fitted candidates and provides tables for comparing information criteria, posterior classification probabilities, class sizes, and convergence diagnostics. Individual models can be extracted with `get_candidate()` for further examination and analysis.

Usage

```
enumerate_classes(
  data,
  vars,
  id,
  n_profiles = 1:4,
  n_group_classes = 1:3,
  model = "basic",
  seed = NULL,
  family = "profiles",
  between_variance = c("varying", "equal"),
  time = NULL,
  ...,
  cores = 1L,
  type = NULL,
  verbose = cores > 1L
)
```

Arguments

data	A data frame with one row per observation and columns containing the indicators, group identifier and, when applicable, occasion order and covariates. Use the same observations and measurement scale throughout a candidate comparison.
vars	Character vector naming the indicator columns in data. Indicator distributions are determined by type and the measurement arguments supplied through ....
id	The column identifying the groups the observations are nested in (students in schools, reports in students). Required; NULL enumerates single-level models, with a message. enumerate_lpa() and enumerate_lca() fit single-level models directly.
n_profiles	Positive integer profile counts to try. Default 1:4.
n_group_classes	Positive integer group-class counts to try. Default 1:3. With id = NULL it is 1 and may be left out.
model	The covariance structures to cross with the class counts. "basic" (the default) fits four structures: "EEI" (equal variances, no covariances), "VVI" (varying variances, no covariances), "EEE" (one covariance matrix shared by all profiles) and "VVV" (a covariance matrix for each profile). Omitting covariances between correlated indicators tends to favour additional profiles. "all" fits all 14 structures of Celeux and Govaert (1995). Otherwise, name structures by their three-letter codes, as multilpa()'s model takes them: the letters give the volume, shape and orientation of each profile's covariance matrix, each equal (E) or varying (V) across profiles, with I for no covariances. NULL crosses no structures. With one continuous indicator, codes that coincide (equal or varying variance) are fitted once; with only categorical indicators no structure is crossed. A structure set through variance_model, covariance_model, volume, shape or orientation in ... replaces the default.
seed	Optional integer random seed accepted by set.seed(). Used to reproduce the starting values for the same data and fitting controls. Invalid values raise latents_bad_argument before fitting begins.
family	"profiles" (the default) enumerates the profile model over n_profiles, n_group_classes and model. One or more of the group-class families "additive", "dispersion" and "additive_dispersion" enumerates those instead, over family, n_group_classes and between_variance; these need id, n_profiles and model are then rejected, and the result is a latents_family_enumeration, read with get_results() and get_candidate().
between_variance	For group-class families: the between-variance restrictions to cross, "varying", "equal" or both (the default). The dispersion family is always fitted with "equal". Refused for the profile family.
time	The occasion column. Enumerates latent transition models (lta()) over n_profiles (default 2:4) and n_group_classes (default 1), with every other lta() argument given in ... held fixed. model, when given, crosses those covariance structures. Only the profile family is available. The result is a latents_transition_enumeration, read with get_results() and get_candidate().

`...` Additional fitting arguments passed to `multilpa()`, or to `lta()` when `time` is supplied. The same settings are used for each candidate. Common arguments include `n_starts`, `max_iter`, `missing`, `categorical`, `centering`, `scaling`, `drop_empty` and `boundary`. The group-class families accept only `n_starts`, `max_iter`, `tol` and `min_variance` through `....`

`cores` Positive integer maximum number of parallel workers; the default, `1`, fits models serially. With a fixed `seed`, parallel and serial execution produce the same results. Parallel fitting requires additional memory. See *Parallel estimation* for platform requirements and the handling of interrupted workers.

`type` Optional model type: `"lca"`, `"lpa"` or `"lta"`. `NULL` (the default) determines the model from the other arguments. `"lca"` treats every indicator as categorical, as `multilca()` does: indicators named in `ordinal` or `count` keep those distributions, and `categorical` adds to the set rather than replacing it. Numeric response codes are categorical with `type = "lca"`. `"lpa"` requires continuous indicators and rejects non-empty `categorical`, `ordinal` or `count`. Both require `time = NULL`. `"lta"` requires `time` and takes its measurement settings from `categorical`, as `lta()` does; it does not make numeric indicators categorical. An explicit type requires `family = "profiles"`; leave `type = NULL` for group-class families.

`verbose` Whether to report progress: one message each when the candidates are prepared, when their starts are fitted, and when the grid is done, with the elapsed time. The default reports when `cores > 1`, where a grid is usually long. Group-class families (`family`) are not reported.

Value

An object of class `multilpa_enumeration` (or the transition or family enumeration described above). Its `type` records the resolved model: `"lca"`, `"lpa"` or `"lta"`, or `"mixed"`/`"family"` for those grids under `type = NULL`. The print header reports this type. `as.data.frame()` gives one row per candidate, starting with its name (`"profiles3_groups2"`), with every criterion and diagnostic (including both entropies and, at each level, the smallest, largest and mean average posterior probability and the number of classes below a cutoff) and the candidate's warnings and error. `get_results(x, "best")` names the candidate at each criterion's minimum; `get_results(x, "classification")` gives every class's average posterior probability and relative entropy for every candidate; `get_results(x, "deltas")` and `"knees"` give model-selection evidence. `summary()` prints each criterion's minimum and every table, `plot()` draws the information criteria (and, with `what`, classification quality and model-selection views), `get_candidate()` extracts a fitted candidate, and `compare_models()` compares candidates selected by name.

Model selection

Information criteria summarize relative support for the candidate specifications, whereas entropy and posterior membership probabilities describe classification uncertainty within each fitted model. These measures address different aspects of the analysis. Model selection should also examine whether the selected likelihood is replicated across starts, whether estimates approach parameter bounds, and whether additional classes yield interpretable differences in response patterns. The function reports this evidence without prescribing a final number of classes.

Comparisons of class counts depend on the measurement specification. For example, additional profiles can represent residual associations that a diagonal covariance model cannot accommodate within profiles. Evaluate the number of classes alongside plausible covariance structures and indicator distributions. The candidate range and fitting controls are supplied by the analyst; the function does not expand the range automatically.

Parallel estimation

Socket workers are used on Windows and on macOS with Apple's Accelerate BLAS; these workers load the installed package, which must match the version used in the main R session. Other supported Unix configurations use forked processes. Each worker requires memory for its fitting tasks.

Depending on the model family, parallel tasks consist of individual starts or complete candidate models. The total number of workers does not exceed `cores`. Warnings and messages are reported in candidate order. Without a fixed `seed`, starting values can differ between runs.

Lost worker results are retried once with fewer workers, and a `latents_worker_lost` warning reports the interruption. Unrecovered tasks are recorded as failed starts or candidates; other results are retained.

Conditions

`latents_bad_argument` when `model` names an unknown structure, when it is given together with another way of setting the structure, or when it is given although no indicator is continuous; when `seed` is not a seed `set.seed()` accepts; when `...` contains an argument `multilpa()` does not take; when `n_group_classes` other than 1 is given with `id = NULL`; when `id` is missing; when `between_variance` is given for the profile family; when `n_profiles` or `model` is given for a group-class family, or `id` is `NULL`; when `time` is combined with a group-class family or with `between_variance`; and when `cores` is not an integer of at least 1; and when `type` is invalid or conflicts with `time`, `family` or the requested measurement distributions.

References

Celeux, G., & Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5), 781–793.

See Also

`get_candidate()` to extract a fitted candidate, `summary()` for the criterion-by-criterion comparison.

Examples

```
set.seed(1)
d <- data.frame(g = rep(1:10, each = 10), y = rnorm(100))
candidates <- enumerate_classes(d, "y", "g", n_profiles = 1:2,
                                n_group_classes = 1, n_starts = 2, seed = 1)
as.data.frame(candidates)
summary(candidates)

# Numeric response codes, explicitly categorical:
```

```
codes <- data.frame(g = rep(seq_len(10), each = 10), item = rep(0:1, 50))
classes <- enumerate_classes(codes, "item", "g", type = "lca",
                             n_profiles = 1, n_group_classes = 1,
                             n_starts = 1, seed = 1)

# Single level, crossing class counts with two named structures:
single <- enumerate_lpa(iris, c("Sepal.Length", "Sepal.Width",
                                "Petal.Length", "Petal.Width"),
                        n_profiles = 1:3, model = c("VVV", "EEE"),
                        n_starts = 2, seed = 1)
summary(single)
```

`enumerate_lca` *Compare latent class models*

Description

`enumerate_lca()` fits latent class models with different numbers of classes to examine heterogeneity in categorical response patterns. It brings together evidence about fit, classification, and estimation stability to support selection of the number of classes. Each fitted candidate remains available for examination of its class-specific response probabilities and class proportions.

Usage

```
enumerate_lca(data, vars, n_classes = 1:4, ...)
```

Arguments

`data`	A data frame with one row per observation.
`vars`	Names of the categorical indicator columns.
`n_classes`	Numbers of classes to compare, specified as positive integers.
`...`	Further arguments for `multilca()`, such as `missing`, `n_starts`, `seed` and `min_probability`, except `id` and `n_group_classes`. Every indicator is categorical unless named in `ordinal` or `count`. `categorical` is accepted but has no further effect; its names must be in `vars`.

Details

This is the single-level interface to `enumerate_classes()` for categorical indicators. Every indicator is categorical unless named in `ordinal` or `count`, as in `lca()`.

For observations nested in groups, use `enumerate_classes()` with the grouping column as `id` and the indicators as `categorical`.

Value

An object of class `multilpa_enumeration`, as `enumerate_classes()` returns: `as.data.frame()` gives one row per candidate, `summary()` the candidate minimizing each criterion, `plot()` the criteria against the number of classes, and `get_candidate()` one fitted candidate. The number of classes is reported in the `n_profiles` column of the enumeration table and selected through the `n_profiles` argument of `get_candidate()`.

Conditions

`latents_bad_argument` when `id` or `n_group_classes` is passed, or when `model` is passed (no indicator is continuous); `latents_bad_categorical` when `categorical` names a column not in `vars`.

See Also

`lca()` to fit one model; `enumerate_lpa()` for continuous indicators.

Examples

```
models <- enumerate_lca(student_esm, c("time_with_friends",
                                       "on_social_media", "sports"),
                        n_classes = 1:3, n_starts = 2, seed = 1)
summary(models)
```

`enumerate_lpa` *Compare latent profile models*

Description

`enumerate_lpa()` fits latent profile models with different numbers of profiles and specified covariance structures. It brings together evidence about fit, classification, and estimation stability to support selection of the number of profiles and their covariance specification. Each fitted candidate remains available for detailed examination.

Usage

```
enumerate_lpa(data, vars, n_profiles = 1:4, model = "basic", ...)
```

Arguments

`data`	A data frame with one row per observation.
`vars`	Names of the indicator columns.
`n_profiles`	Numbers of profiles to compare, specified as positive integers.
`model`	The covariance structures: `"basic"`, the default, for the four that combine equal or varying variances with covariances absent or present (`EEI`, `VVI`, `EEE`, `VVV`); `"all"` for all 14; or their three-letter codes. See `enumerate_classes()`.
`...`	Further arguments for `multilpa()`, such as `missing`, `n_starts`, `seed`, `boundary` and `categorical`, except `id` and `n_group_classes`.

Details

This is the single-level interface to `enumerate_classes()`. For observations nested in groups, use `enumerate_classes()` with the grouping column as `id`, which also varies the number of group classes.

Value

An object of class `multilpa_enumeration`, as `enumerate_classes()` returns: `as.data.frame()` gives one row per candidate, `summary()` the candidate minimizing each criterion, `plot()` the criteria against the number of profiles, and `get_candidate()` one fitted candidate.

Conditions

`latents_bad_argument` when `id` or `n_group_classes` is passed, or when `model` names an unknown structure.

References

Celeux, G., & Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5), 781–793.

See Also

`lpa()` to fit one model; `enumerate_lca()` for categorical indicators; `bootstrap_lrt()` to test one number of profiles against another.

Examples

```
models <- enumerate_lpa(srl, c("cognitive_strategies", "intrinsic_value",
                               "self_efficacy", "self_regulation",
                               "test_anxiety"),
                        n_profiles = 1:3, n_starts = 2, seed = 1)
summary(models)
plot(models)
```

---

enumerate_regressions *Assess the number of classes in mixture regression models*

Description

`enumerate_regressions()` compares mixture regression models with different numbers of latent classes. It fits the same outcome model across the requested class counts, allowing researchers to assess whether additional classes provide a useful representation of variation in outcome–predictor relationships. The outcome distribution, predictors, and other supplied model settings remain common across candidates.

For supported two-level models, the function can also vary the number of group classes and fit each requested combination of class counts. The resulting table brings together information criteria, classification measures, class sizes, convergence, and likelihood replication. Individual fitted models are retained for examination of their coefficients, predictions, and class assignments.

An optional parametric bootstrap likelihood-ratio test compares models with consecutive numbers of regression classes while holding the number of group classes fixed. This provides additional evidence about whether the improvement in likelihood supports the more complex specification.

Usage

```
enumerate_regressions(
  formula,
  data,
  n_classes = 1:4,
  n_group_classes = 1L,
  bootstrap = 0L,
  bootstrap_starts = 3L,
  seed = NULL,
  ...
)
```

Arguments

`formula`	A model formula `outcome ~ predictors`. Factors, interactions and `offset()` are supported.
`data`	A data frame.
`n_classes`	Positive integer class counts to fit. Default `1:4`.
`n_group_classes`	Positive integer group-class counts (two-level model; needs `id`), crossed with `n_classes`. Default 1.
`bootstrap`	Number of bootstrap replicates for the likelihood-ratio test; `0` (the default) skips it. Not available with sampling weights.
`bootstrap_starts`	Random starts for each bootstrap refit. Default 3.
`seed`	`NULL` (default) advances the caller's random stream. An integer seed restores the caller's `.Random.seed` afterwards. R does not store a cached Box-Muller normal draw in `.Random.seed`, so that hidden cached draw cannot be preserved.
`...`	Further arguments to `mixture_regression()`, such as `family`, `id`, `class_level`, `common`, `membership` or `variance`. `n_classes`, `n_group_classes` and `vcov_type` are not accepted here.

Details

The optional parametric bootstrap likelihood-ratio test compares `k - 1` with `k` classes at a fixed number of group classes (McLachlan, 1987; Nylund, Asparouhov and Muthen, 2007). Outcomes are simulated under the smaller model, and both models are refitted to each simulated dataset. The

p-value is `(1 + b) / (1 + V)`, where `b` is the number of valid replicate statistics at least as large as the observed statistic and `V` is the number of valid replicates.

A replicate is invalid if a refit fails, does not converge, produces a separation or degenerate random-effect covariance warning, or yields a negative likelihood-ratio statistic beyond numerical tolerance. The p-value uses valid replicates only. Both valid and invalid counts are reported, and `latents_failed_replicates` warns when any replicate is invalid. The p-value is `NA` when fewer than half of the requested replicates are valid. Excluding invalid replicates conditions the reference distribution on successful estimation and may affect calibration. Report the invalid proportion and examine the refit diagnostics when interpreting the test.

Value

An object of class `latents_regression_enumeration`. Its table (from `as.data.frame()` or `get_results(x, "fit")`) has one row per model, with columns `n_classes`, `n_group_classes`, `log_likelihood`, `n_parameters`, `aic`, `bic`, `bic_rows`, `sabic`, `icl`, `entropy`, `smallest_share`, `converged`, `n_best_replicated`, `note` (why a model could not be fitted), `best_bic` (`TRUE` for rows attaining the smallest reported BIC, without filtering on convergence), and, with `bootstrap > 0`, `blrt_statistic`, `blrt_p_value`, `blrt_replicates` (the valid replicates), `blrt_n_invalid` (the failed, unconverged, separated or reversed-likelihood replicates) and `blrt_flagged` (a compatibility alias for `blrt_n_invalid`). The test columns are `NA` where the `k - 1` or `k` class model is missing or failed, and the p-value is `NA` when the observed fits fail validation or fewer than half of the requested replicates are valid. `get_results(x, "model", n_classes = , n_group_classes = )` returns one fitted model, `get_results(x, "tree")` the class tree and `get_results(x, "all")` the named list of the `fit` and `tree` tables; any other `what` raises `latents_bad_argument`. `print()` shows the main columns of the table. `summary()` returns the same table as `as.data.frame()`. `plot()` draws AIC, BIC, SABIC and ICL across the classes (see `plot_enumeration()` for its other views, and `plot_views(x)` for the views a regression grid offers).

Interpretation

The `best_bic` column identifies the smallest reported BIC among the fitted candidates; it does not exclude unconverged fits. Model assessment should therefore consider convergence and likelihood replication alongside information criteria, class sizes, and the meaning of the estimated regression relationships. Bootstrap results also require attention to unsuccessful refits, whose exclusion can affect the test.

Conditions

A model whose fit raises `latents_no_valid_start` or `latents_bad_data` at some count has a row of `NA` values with the condition message in `note` (and `print()` says how many); when every model fails with the same `latents_bad_data` error (a column that does not exist), that error is raised instead. Other errors stop the enumeration. `latents_bad_argument` for invalid `n_classes`, `n_group_classes`, `bootstrap` or `bootstrap_starts`, for an invalid `seed`, for an argument in `...` that `mixture_regression()` does not have, for `n_classes`, `n_group_classes` or `vcov_type` passed through `...`, for a bootstrap test whose outcome is not a plain column name, and from `get_results()` for a model that is not in the grid or could not be fitted. `latents_unsupported_weights` for a bootstrap test with sampling weights. `latents_failed_replicates` is warned, naming the count, when some replicates are invalid (the p-value then uses the valid ones), and when the observed fits fail validation or fewer than half of the replicates are valid (the p-value is then `NA`).

References

McLachlan, G. J. (1987). On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Applied Statistics*, 36, 318–324.

Nylund, K. L., Asparouhov, T., & Muthen, 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, 535–569.

Examples

```
classes <- enumerate_regressions(score ~ hours, data = study_hours,
                                 n_classes = 1:3, n_starts = 3, seed = 1)
classes

# A small bootstrap count is used only to limit example execution time.
with_test <- enumerate_regressions(score ~ hours, data = study_hours,
                                   n_classes = 1:2, n_starts = 3, seed = 1,
                                   bootstrap = 4)
as.data.frame(with_test)
```

`fit_staged` *Estimate group classes with fixed individual profile definitions*

Description

`fit_staged()` estimates a multilevel mixture model in which observed groups are distinguished by their composition of individual profiles. It separates the estimation of profile definitions from the estimation of group classes, allowing between-group differences to be studied while retaining a common measurement model.

The first stage estimates the individual profiles using a model with one group class. These profiles are defined by their indicator distributions, including means and covariance parameters for continuous indicators and response probabilities for categorical indicators. The second stage holds these measurement parameters fixed and estimates the profile probabilities within each group class, together with the proportions of the group classes. Group classes therefore differ in their profile composition while the profiles retain the same measurement definitions.

A compatible fitted measurement model may also be supplied, allowing established profile definitions to be reused. The returned model contains the second-stage estimates and retains the first-stage fit, so both stages can be inspected.

Usage

```
fit_staged(
  data,
  vars,
  id,
  n_profiles,
```

```
  n_group_classes = 2L,
  measurement = NULL,
  variance_model = c("varying", "equal"),
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  min_variance = NULL,
  seed = NULL,
  missing = c("error", "fiml"),
  covariance_model = c("diagonal", "full"),
  categorical = character(),
  min_probability = 1e-10,
  time = NULL,
  search = NULL,
  n_screen = NULL
)
```

Arguments

`data`	A data frame containing the indicators and the group identifier.
`vars`	Unique character vector of indicator column names. Indicators named in `categorical` use categorical response distributions; the remaining indicators are continuous and use Gaussian distributions.
`id`	Name of the observed group identifier column. Character, factor and numeric identifiers are supported; missing identifiers are not. Required; `id = NULL` is not accepted.
`n_profiles`	Positive integer number of individual profiles.
`n_group_classes`	Number of latent group classes in the second stage. Must be at least two; the default is 2.
`measurement`	An optional fitted `multilpa` model supplying the first-stage measurement estimates. With `NULL` (the default), the first stage is estimated from `data`. A supplied model must have one group class and matching profiles, indicators and measurement options. Models with `weights`, `centering`, `scaling`, `ordinal` or `count` indicators, `direct_effects`, or `local_dependence` between continuous indicators are not supported. Supported covariance structures are EEI, VVI, EEE and VVV. An incompatible measurement model raises `latents_bad_stage`.
`variance_model`	`"varying"` (the default; profile-specific variances or covariance matrices) or `"equal"` (shared across profiles).
`n_starts`	Number of EM starts used in each estimated stage. When `measurement` is supplied, only the second stage is estimated.
`max_iter`	Nonnegative integer maximum number of EM updates per start in each estimated stage (default 1000). Zero evaluates initial values without performing EM updates.
`tol`	Positive convergence tolerance (default `1e-8`). EM has converged when one step (or SQUAREM cycle) changes the log likelihood by no more than `tol` times the

number of observed indicator values (each non-missing cell in a row of positive weight, with a minimum count of one). The tolerance is therefore expressed per observed indicator value.

`min_variance` Lower bound on each continuous-indicator variance, or on each covariance eigenvalue under full covariance. `NULL` (the default) uses `1e-6` times the smallest observed sample variance of the continuous indicators. A numeric value specifies an absolute bound in squared indicator units. Estimates at the bound raise a warning. In the second stage, measurement parameters are held at their first-stage values.

`seed` Optional random seed: any integer `set.seed()` accepts, including negative integers. When supplied, the caller's random-number state is restored on exit.

`missing` `"error"` (the default) rejects missing indicators; `"fiml"` maximises the observed-data likelihood under an ignorable missingness mechanism (MAR), integrating over missing indicators.

`covariance_model`
`"diagonal"` (the default) assumes conditional independence; `"full"` estimates within-profile residual covariances.

`categorical` Character vector naming indicators to treat as categorical. Each is modelled by unrestricted, profile-specific response probabilities over its observed categories (the latent class measurement model); binary, ordinal and unordered indicators are handled by the same parameterization. Numeric, integer, logical, character and factor columns are accepted. Indicators not named here stay Gaussian, so naming a subset fits a mixed-mode model.

`min_probability`
Positive lower bound on every categorical response probability (default `1e-10`), constraining the maximum-likelihood problem as `min_variance` does for Gaussian indicators.

`time` Optional name of a column giving each observation's position within its group, such as a wave, occasion or course number. It is not used in estimation; it is stored for `get_results(x, "sequences")`, `get_results(x, "sequence_summary")` and `plot(what = "sequences")`. Values must be complete and unique within each group, and are ordered as `sort()` orders them: numbers numerically, a factor by its levels, and character strings alphabetically. Numbers stored as text whose alphabetical order is not their numeric order (`"2"`, `"10"`), and numbered labels sharing their text around the number whose alphabetical order is not the numbers' (`"T2"`, `"T10"`; `"week 2"`, `"week 10"`), raise `latents_bad_time` (also `latents_bad_data`); give the order with `factor(levels = )` or zero-pad the numbers. Zero-padded numbers and ISO dates are accepted.

`search` Initialisation strategy for each estimated stage: `NULL` (the default), `"kmeans"` or `"emEM"`. The default is `"emEM"` for models containing continuous indicators with a shared covariance matrix (EEI or EEE), and `"kmeans"` otherwise. `"kmeans"` uses clustering-based starts; `"emEM"` screens a pool of starts with short EM runs before completing `n_starts` selected searches. The second stage uses the first-stage measurement estimates as supplied starting values and holds them fixed. See `multilpa()` for the screening and start-selection procedure.

`n_screen` Number of starts screened under `search = "emEM"`, at least `n_starts`. The default is five times `n_starts`. Screening is skipped in the second stage when

`n_starts = 1`, because a starting measurement solution is supplied. A value with `search = "kmeans"`, or one below `n_starts`, raises `latents_bad_argument`.

Details

Information criteria and likelihood-ratio comparisons condition on the fixed measurement model.

With `parameter_inference(fit, method = "bootstrap")`, groups are resampled and both stages are re-estimated. This procedure propagates first-stage estimation uncertainty and reports standard errors and intervals for the measurement parameters as well as the second-stage parameters. A supplied `measurement` model is re-estimated in each resample using its fitting options.

For staged models with the same fixed measurement specification, use `n_parameters` when comparing conditional information criteria. Comparisons with jointly estimated models can include the measurement parameters through `n_parameters_with_measurement`, but remain descriptive because the staged estimates do not generally maximize the joint likelihood.

Value

A second-stage `multilpa` object with measurement parameters equal to the first-stage estimates. The `fixed` component identifies the parameter blocks held constant. `n_parameters` counts the parameters estimated in the second stage; `n_parameters_with_measurement` also includes the first-stage measurement parameters. `get_results(fit, what = "stages")` returns a two-row summary of the estimation stages. Standard result-extraction and diagnostic methods for `multilpa` objects are available.

Interpretation

The second-stage estimates are conditional on the fixed measurement model. Default Wald standard errors omit uncertainty from estimating that model; bootstrap inference can propagate this uncertainty by resampling groups and re-estimating both stages. Staged estimates do not generally maximise the joint likelihood, so comparisons with jointly estimated models require care.

See Also

`multilpa()` for selective constraints through `fixed`; `starting_values()` with `what = "measurement"` to extract measurement parameters for a subsequent fit.

Examples

```
set.seed(7)
example_data <- data.frame(
  school = rep(seq_len(30), each = 8),
  score_a = stats::rnorm(240), score_b = stats::rnorm(240)
)
staged <- fit_staged(example_data, c("score_a", "score_b"), "school",
                     n_profiles = 2, n_group_classes = 2, n_starts = 2,
                     seed = 1)
get_results(staged, what = "stages")
get_results(staged, what = "profile_probabilities")
```

get_candidate *Retrieve a fitted model from an enumeration or comparison*

Description

`get_candidate()` retrieves a specified fitted model from an enumeration grid or model comparison for detailed inspection and subsequent analysis. The returned object retains its original model class and supports the summary, diagnostic, plotting, and inference methods available for that model.

A candidate can be identified by its name or, for supported enumeration grids, by its class counts and model specification. When several candidates share the same class counts, the covariance structure or candidate name distinguishes them. The function retrieves the stored fit without repeating estimation.

Usage

```
get_candidate(x, n_profiles, n_group_classes = 1L, model = NULL)

candidate_fit(x, n_profiles, n_group_classes = 1L, model = NULL)
```

Arguments

`x`	An enumeration grid from `enumerate_classes()` (a profile, latent class, transition or group-class family grid), or a comparison from `compare_models()`. On a family grid, give the name, or `n_group_classes` with `model`. A grid of mixture regressions from `enumerate_regressions()` is not supported (`latents_bad_argument`): `get_results(grid, "model", n_classes = )` returns one of its models.
`n_profiles`	Number of individual profiles of the candidate, the candidate's name (`"profiles3_groups4"`), or its two class counts (`c(3, 4)`). For a comparison, the row label, as its `model` column shows it.
`n_group_classes`	Number of group classes of the candidate. Default 1.
`model`	The covariance structure of the candidate, needed when `enumerate_classes()` crossed several structures and the class counts match several candidates; on a group-class family grid, the family model as the grid's `model` column shows it.

Details

A candidate is selected by its name, as the grid's `candidate` column shows it (`get_candidate(grid, "profiles3_groups4")`; `classes` instead of `profiles` when every indicator is categorical); by its class counts, `get_candidate(grid, 3, 4)` or `get_candidate(grid, c(3, 4))`; or, on a grid that crosses covariance structures, by its counts and `model`. From a comparison, give the row label: `get_candidate(comparison, "profiles3_groups4")`.

`candidate_fit()` is the former name of `get_candidate()`, kept for compatibility. It takes the same arguments, forwards to `get_candidate()` and gives no warning.

Value

The fitted object for the selected candidate, with the class and methods of its fitting function (for example, `multilpa` for a profile model).

Conditions

`latents_unknown_candidate` for a name or class counts not in the grid, for counts that match several candidates (the message lists their names), and for a label that is not a row of a comparison; `latents_failed_candidate` when the candidate's fit raised an error, so there is no model to return; `latents_bad_argument` for a name or pair of counts given together with `n_group_classes` or `model`, for a value that names no candidate, and for a comparison given no label; `latents_incomplete_fit` for a comparison that no longer carries its models. Counts that are not single positive integers fail the argument checks.

Examples

```
set.seed(1)
d <- data.frame(g = rep(1:10, each = 10), y = rnorm(100))
candidates <- enumerate_classes(d, "y", "g", n_profiles = 1:2,
                                n_group_classes = 1, n_starts = 2, seed = 1)
get_candidate(candidates, n_profiles = 2, n_group_classes = 1, model = "VVI")
# This grid crosses two structures, so the names carry them:
candidates
get_candidate(candidates, "profiles2_groups1_VVI")
compared <- compare_models(candidates, "profiles1_groups1_VVI",
                           "profiles2_groups1_VVI")
get_candidate(compared, "profiles2_groups1_VVI")
```

---

get_group_tna *Construct transition networks for latent group classes*

Description

`get_group_tna()` constructs separate transition networks for latent group classes, allowing patterns of persistence and movement between states to be compared. Each network represents the same latent profiles or classes, making differences in transition probabilities directly comparable.

For a supported latent transition model, each network uses the estimated transition matrix and initial state distribution of its group class. For a profile or class model fitted with `time`, it summarises the modal state sequences of observed groups assigned to each latent group class. The returned object supports grouped network analysis and visualisation with the tna package.

Usage

```
get_group_tna(x, ...)

## S3 method for class 'multilpa_transitions'
get_group_tna(x, label = "Group class", ...)
```

Arguments

x	A supported time-homogeneous model from lta(), or a multilpa() or multilca() fit made with time, whose modal profile sequences are then used.
...	Passed to tna::tna() for each class.
label	Name given to the classes in tna's output (default "Group class").

Details

For a transition-matrix row with no expected outgoing transitions, the fitted probabilities are retained and a latents_empty_transition_row warning identifies the row as unestimated. General lta() models are currently unsupported; see get_tna() for the supported model scope.

Value

An object of class group_tna, a list of tna models, one per latent group class. centralities() returns one table with a group column, compare() contrasts two classes, and plot() draws all networks.

Interpretation

Networks represent latent group classes, rather than individual observed groups. Networks constructed from modal sequences treat both state and group-class assignments as known and do not account for uncertainty in either classification.

See Also

get_tna() for one network over the whole sample.

Examples

```
if (requireNamespace("tna", quietly = TRUE)) {
  activity <- c("browse", "lectures", "forum_read")
  moves <- lta(course_engagement, vars = activity, id = "student",
                           time = "sequence", n_profiles = 2,
                           n_group_classes = 2, n_starts = 2, seed = 1)
  get_group_tna(moves)
}
```

---

get_results *Extract model estimates and analysis results*

Description

`get_results()` provides access to the estimates and summaries produced by models and analyses in `latents`. It supports examination of the estimated latent classes, their membership probabilities, and model-fit statistics, as well as the preparation of results for reporting and further analysis.

The results available depend on the supplied object. Fitted models provide measurement estimates, class proportions, posterior membership probabilities, and other model-specific quantities, such as transition probabilities. Enumeration and comparison objects provide summaries of candidate models, while diagnostic and bootstrap objects provide their corresponding checks and test results.

The `what` argument selects the requested table. Each table is returned as a standard R data frame, allowing its contents to be inspected, filtered, combined, or exported. Omitting `what` returns the object's primary table; `what = "all"` returns a named collection of the tables included by default for that object.

Usage

```
get_results(x, what = NULL, ...)
```

Arguments

`x`	A fitted model of this package, an enumeration grid, a model comparison, a bootstrap comparison, a starting-value set, a classification-diagnostics object, or the summary of any of them.
`what`	Name of the table to return, or `"all"` for the collection of tables included by default for the object. `NULL` (default) returns the primary table: the measurement model for a fit, the candidate grid for an enumeration, the test row for a bootstrap comparison and the entropy table for a diagnostics object. A name the object does not offer raises `latents_bad_argument`, listing the names it does offer.
`...`	Arguments for the requested table. Each table accepts only the arguments listed for it under *Tables*; any other argument raises `latents_bad_argument`.

Details

Some tables calculate additional quantities when requested. For example, measurement tables may obtain standard errors through `parameter_inference()`. Their availability therefore depends on the fitted model and the requirements of the corresponding calculation.

The table catalogue below specifies the contents and accepted arguments of each table, including results that require an explicit request.

Every object also has `print()`, which shows a compact overview, `summary()`, which collects its main tables (taking `level` and `vcov_type` where the model has standard errors), and `as.data.frame()`, which returns its primary table, the same as `get_results(x)`.

Before combining supplied `data` with model results, the function checks row alignment using columns retained by the fitted object, such as identifiers, occasions, and indicators. Numeric values serialized by `utils::write.csv()` are also accepted when serialization preserves distinct values. Columns absent from the fitted object cannot establish alignment. Inference and model comparison apply additional checks to verify that supplied data reproduce the estimation data.

Value

A base data.frame for a single what, whose columns are given under *Tables*, or a named list of such data frames for what = "all".

Tables

The following tables apply to profile and class models and, where supported, membership-covariate and transition models. Tables for other model families are described in *Tables of the other model families*.

"profiles" The Gaussian measurement model, one row per profile and continuous indicator: profile, indicator, mean, variance, standard_deviation, mean_standard_error, variance_standard_error. Zero rows, with the same columns, when every indicator is categorical. Takes scale and data.

"responses" The categorical measurement model, one row per profile, categorical indicator and category: profile, indicator, category, probability, threshold, probability_standard_error. threshold is qlogis(P(y <= category)) and is NA_real_ for each indicator's final category; a probability on its bound has an NA_real_ standard error. Zero rows, with the same columns, when no indicator is categorical. Takes data.

"covariances" The within-profile residual covariance matrices, one row per profile and ordered pair of continuous indicators: profile, indicator, indicator_2, covariance. Under the diagonal parameterization the off-diagonal covariances are reported as exact zeros.

"ordinal", "count_means" Fits with ordinal or count indicators only: the ordinal and count measurement models. Take data.

"counts" Effective class sizes, one row per class at each level of the fit: level, class, effective_count (the summed posteriors) and effective_proportion. Effective counts use posterior probabilities; modal counts in "classification" use discrete class assignments.

"model" One row describing the fit: its dimensions, likelihood, parameter counts, aic, bic_groups, bic_individual and its convergence and boundary diagnostics. The columns differ by family (a covariate fit adds its predictor counts); "information_criteria" has the same columns for every family.

"posteriors" One row per individual and profile: row, group, profile, posterior, modal. Takes format. Excluded observations retain their rows, with posterior and modal set to NA. In wide format their assigned profile and posterior columns are NA.

"assignments" One row per observation: the columns the fit was built from (or those of data), then profile, group_class where the model has one, uncertainty (one minus the posterior of the assigned profile) and one posterior_profile_* column per profile, plus posterior_noise for a fit with a noise component (whose cases are profile 0). Takes data and truth; with truth it returns a recovery table instead (see *Recovery*). Row order is preserved. Under the default drop_empty = TRUE, observations with no observed indicators have NA for profile, uncertainty and the posterior columns. Group assignments are missing only when the entire group is excluded; an empty observation can therefore retain a group assignment supported by other observations. Zero-weight units are unclassified under either drop_empty setting.

"classification" Classification quality, one row per level and class: level, class, n_modal, proportion_modal, estimated_n, estimated_proportion, average_posterior, odds_correct_classification. Takes level. Under sampling weights every column that estimates the population is weighted: estimated_n and estimated_proportion are weighted posterior totals and average_posterior

a weighted mean, so the odds of correct classification compare the two on the same population; `n_modal` and `proportion_modal` stay counts of the units supplied. The odds are `Inf` for a class classified perfectly (average posterior one to within rounding) and `NA` where they are undefined: a class with no modal members, or a single class, whose proportion is one.

`"average_posteriors"` For the units modally assigned to each class, their mean posterior for every class: `level`, `assigned_class`, `class`, `n_assigned`, `average_posterior`. Rows sum to one within each assigned class. Under sampling weights `average_posterior` is the weighted mean and `n_assigned` stays a count. This conditions on the assigned class, whereas `"classification_errors"` conditions on the true class. Takes `level`.

`"classification_errors"` The probability that a unit in one true class is assigned to another: `level`, `true_class`, `assigned_class`, `probability`, summing to one within each `true_class`. Takes `level`. It is `sum_i w_i p_it 1(W_i = s) / sum_i w_i p_it` for true class `t` and assigned class `s` (Vermunt 2010), with `w_i` the unit's sampling weight under sampling weights and 1 otherwise, so a weighted fit reports the population's error rates. On a membership-covariate fit this is a marginal summary over the fitted units, not conditional on the covariates. A fit with a noise component raises `latents_unsupported_noise`, as `three_step()` does, and `"all"` and `summary()` omit the table.

`"bch_weights"` The BCH inverse-error weights, one row per unit and class: `level`, `unit`, `assigned_class`, `class`, `weight`. Takes `level`; a weighted regression should use one level, not `"both"`. They invert the `"classification_errors"` matrix, weighted as it is. On a membership-covariate fit the weights invert the marginal error matrix and are not a conditional three-step correction. Not available for a fit with a noise component, as for `"classification_errors"`.

`"entropy"` One row per level: `level`, `n_classes`, `n_units`, `entropy_sum`, `relative_entropy`. Under sampling weights `entropy_sum` is `sum_i w_i e_i` and `relative_entropy` `1 - sum_i w_i e_i / (sum_i w_i l`. `n_units` stays a count.

`"replication"` One row: `n_starts` (starts run to the end), `n_best_replicated` (those whose log likelihood is within `tolerance` of the best), `share`, `tolerance`, `best_start` (the selected start's index) and `log_likelihood`. A best solution reached by one start of several may not be the global maximum: refit with more starts.

`"residuals"` Bivariate residuals for every pair of indicators, checking within-profile independence: `profile`, `indicator_1`, `indicator_2`, `kind` (`"gaussian"`, `"categorical"`, `"mixed"` or `"count"`), `observed`, `expected`, `residual`, `effective_n`, `statistic`, `df`, `p_value`, `p_adjusted`. Takes `data`, `by` and `adjust`. See *Bivariate residuals* for what each kind computes.

`"information_criteria"` Every information criterion the package computes (see *Information criteria*). Takes `format` and `definitions`.

`"sequences"` The assignments in occasion order, one row per classified observation (a row with nothing observed is left out, or kept under `drop_empty = FALSE`): `group`, `group_class`, `time`, `profile`. Takes `format`. Raises `latents_no_time` for a fit made without `time`.

`"sequence_summary"` One row per group class: `group_class`, `groups`, `observations`, `mean_length`, `median_length`, `shortest`, `longest`, `complete`, `gaps`. Summarises classified groups only; lengths describe their recorded occasions, including occasions with missing indicators. Raises `latents_no_time` for a fit made without `time`.

`"starts"` One row per EM start: `start`, `log_likelihood`, `converged`, `iterations`, `error` and, where the family records it, `boundary`. Under `search = "emEM"`, one row per start of the pool, with its `kind`, its `stage` (`"final"` when run to convergence, `"screened"` otherwise) and its `screened_log_likelihood`.

"data" The data the model was fitted to, one row per observation in input order: the identifier (except for a single-level fit), the occasion where there is one, the sampling weights of a weighted fit (as rescaled for fitting) and the indicators under their original names. Columns the model did not use, including membership covariates, are not included; three_step() and r3step() therefore take data.

Fits with discrete group classes also offer:

"group_posteriors" One row per observed group and group class: group, group_size, log_likelihood, group_class, posterior, modal. Takes format. Excluded groups retain their rows, with missing posterior probabilities and modal assignments, as for "posteriors".

Tables specific to one family or object:

"gaussian_diagnostics" Gaussian profile, membership-covariate, transition and cross-level fits that record these diagnostics. One row per Gaussian component and measurement context: context, profile, gaussian_mass, positive_rows, weight_effective_n, min_variance_ratio, min_standardized_eigenvalue, boundary. gaussian_mass sums posterior times normalized sampling weight over rows with at least one observed Gaussian indicator; positive_rows counts those rows (the same for every component, since a Gaussian density is positive everywhere). weight_effective_n is the concentration measure (sum(a)^2 / sum(a^2)) for those contributions, not an identification guarantee; Gaussian mass is not this effective sample size. min_variance_ratio is the smallest component variance divided by its observed sampling-weighted marginal variance on the fitted scale, including between-component variation. min_standardized_eigenvalue is the smallest eigenvalue of the component covariance after standardizing every indicator by its observed sampling-weighted marginal standard deviation. Missing reference variation gives NA ratios. boundary concerns the absolute variance or eigenvalue floor. These descriptive measures use no automatic spike cutoff and do not alter start selection.

"profile_probabilities", "stages" A multilpa fit. "profile_probabilities" has one row per group class and profile: group_class, profile, probability, group_class_probability. "stages" has one row per estimation stage: stage, group_classes, fixed, log_likelihood, parameters, parameters_with_measurement, converged. An ordinary fit has one "joint" row and a fit_staged() result has two; fixed is NA_character_ for a stage that estimated every block.

"coefficients" A covariate fit. One row per estimated membership coefficient: level ("profile" or "group"), outcome, term, parameter (always "logit"), estimate. Standard errors are given by parameter_inference().

"transitions", "initial", "sequence_lengths" A transition fit. "transitions" has one row per group class and ordered pair of profiles: group_class, from, to, probability, expected_count, stable, estimated, group_class_probability; it takes estimated and stable. "initial" has one row per group class and profile: group_class, profile, probability, prevalence, group_class_probability. "sequence_lengths" has one row per group: group, group_class, occasions, observations, complete.

"candidates", "best", "classification" An enumeration grid. "candidates" has one row per candidate model: its name (candidate, first), its class counts, covariance model, log likelihood, parameter count, every criterion under both sample-size conventions, both entropies, the average posterior probability (AvePP) summaries of each level (avepp_min, avepp_max,

avepp_mean, avepp_below and the same four prefixed group_) and the convergence, boundary, replication, warning and error diagnostics, including min_standardized_eigenvalue and min_gaussian_mass where Gaussian diagnostics are available. These are descriptive summaries, without an automatic cutoff or change to candidate selection. Failed candidates are kept with NA estimates and their error text. It takes cutoff (default 0.8): avepp_below and group_avepp_below count the classes whose AvePP is below it. A class with no modally assigned unit has no AvePP; it is excluded from the minimum, maximum and mean and counted as below the cutoff. A level with one class has NA summaries. "best" has one row per information criterion and sample-size convention: criterion, convention, candidate (the converged candidate with the lowest value), n_profiles, n_group_classes, model and value; what = "criteria" is accepted as a deprecated name for it. "classification" has one row per fitted candidate, level and class: candidate, n_profiles, n_group_classes, model, then the columns of a fit's "classification" table and that level's relative_entropy. Failed candidates and single-class levels contribute no rows.

"deltas", "knees" Model-selection evidence of an enumeration grid, each taking using (default "bic_groups"). Neither selects a model. "deltas" has one row per candidate: candidate, its class counts and model, criterion, value, delta_profile (the decrease in the criterion from the converged candidate with one profile fewer, the same group classes and the same structure; positive means this candidate is lower), from_profile naming that candidate, and delta_group_class with from_group_class for one group class fewer. value and both deltas are NA for a candidate that did not converge, and a delta is NA when the smaller neighbour is missing or did not converge. "knees" has one row per line of the grid: direction ("profiles", along the profiles at fixed group classes; "group_classes", along the group classes at fixed profiles; "parameters", every candidate by its number of parameters, one line per structure), line, criterion, from, to, n_points, knee_candidate, distance, knee_without_largest (the knee when the last candidate of the line is left out), stable and note. The knee is the candidate furthest below the straight line from the first to the last candidate, measured perpendicular to it with both axes rescaled to 0-1. This is a simplified rule inspired by Kneedle (Satopaa et al., 2011), without its smoothing or sensitivity thresholds, so a very small positive distance can still name a knee. A line needs three candidates for a knee and four for knee_without_largest and stable; otherwise they are NA and note gives the reason.

"local_dependence" For a fit with local_dependence: pair, kind ("categorical" or "continuous"), profile, item_1, category_1, item_2, category_2, probability, independent, ratio, covariance and correlation. A categorical pair has one row per profile and category combination, including unobserved combinations subject to the min_probability floor and included in the parameter count: the fitted joint probability, the product of the two marginal probabilities (independent, the joint probability under independence within the profile) and their ratio; its covariance columns are NA. A continuous pair has one row per profile, with the pair's covariance and the correlation it implies; its category and probability columns are NA. A fit without pairs raises latents_no_local_dependence, and what = "all" leaves the table out.

"direct_effects" For a fit with direct_effects, one row per effect: indicator, covariate, estimate (the change in the indicator's mean per unit of the covariate, the same in every profile), standard_error, statistic, p_value, conf_low and conf_high (Wald, from parameter_inference()). A fit without direct effects raises latents_no_direct_effects, and what = "all" leaves the table out.

"lmr" The Vuong-Lo-Mendell-Rubin (VLMR) and adjusted Lo-Mendell-Rubin (aLMR) tests of

an enumeration grid (Vuong 1989; Lo, Mendell and Rubin 2001; Vermunt 2024), one row per pair of converged candidates that differ by one profile (`direction = "profiles"`) or one group class (`"group_classes"`) at the same structure: `direction`, `null_candidate`, `alternative_candidate`, `n_profiles`, `n_group_classes` and `model` of the alternative, `n_units` (the independent units scored: groups, or observations without groups), `parameters_added`, `lr` (twice the log-likelihood difference), `vlmr_p`, `adjusted_lr` (`lr / (1 + 1 / (parameters_added * log(n_units)))`), `almr_p`, `reference_mean` and `reference_sd` of the weighted chi-square reference distribution, whose weights are the eigenvalues of Vuong's matrix of the two models' scores and Hessians, and `note`. The p-values are computed by Imhof's method to about `1e-9`. A pair whose information matrix is singular, or whose candidate is on a boundary, keeps its row with `NA` statistics and the reason in `note`; warnings raised while scoring a candidate are recorded there too. Every candidate's information is computed, so the table is built only when asked for, and `what = "all"` leaves it out. A grid with no such pair raises `latents_no_lmr`.

`"tree"` The flows of `plot_tree()`, on every grid, taking its arguments `n_group_classes`, `n_profiles` (`n_classes` on a regression grid), `model`, `candidates`, `from`, `to` and `level` (default `"individuals"`). One row per pair of classes in consecutive rows of each tree: `tree` (the tree's name, as in the plot list), `level`, `from_candidate`, `from_class`, `from_share`, `to_candidate`, `to_class`, `to_share`, `mass` (the share of all units the two classes share, the sum over units of the product of their posterior probabilities, weighted by sampling weights at the individual level), `share_of_from` and `share_of_to` (`mass` as a part of each class). Every pair is listed, including those that share no mass. Units are the rows and groups the fit counts (see `drop_empty`), weighted by their sampling weights at both levels. A grid with no tree raises `latents_no_tree`, a fit with a noise component `latents_unsupported_noise`, and `what = "all"` leaves the table out. The summary of a grid carries the default table only, without these arguments.

`"comparison"`, `"classification"` A `compare_models()` result. `"comparison"` is its table, one row per model. `"classification"`, offered when every model is a profile, class, membership-covariate or transition model, has one row per model, level and class: `model`, then the columns of a fit's `"classification"` table and that level's `relative_entropy`; single-class levels are left out. The models themselves are returned by `get_candidate()`.

`"test"`, `"replicates"` A bootstrap comparison. `"test"` is the one-row result; `"replicates"` has one row per simulated dataset.

`"profiles"`, `"responses"`, `"profile_probabilities"`, `"covariances"` A starting-value set: its Gaussian means and variances, response probabilities, profile probabilities and, one row per profile and ordered pair of continuous indicators, its covariances (`profile`, `indicator`, `indicator_2`, `covariance`).

`"entropy"`, `"classification"`, `"average_posteriors"`, `"residuals"`, `"replication"` A `diagnostics()` object: the tables of the same names described above.

Tables of the other model families

Each family's first table is its default `what`; `"all"` returns a named list of every table (without `"recovery"`, which needs `data` and `truth`). Tables with intervals take `level`, and `vcov_type` (`"observed"`, `"robust"` or `"opg"`; `NULL` reuses the type stored with a regression fit). These objects also have `coef()`, `vcov(type = )`, `confint(parm = , level = )`, `logLik()` and `nobs()` methods. `confint()` of a mixture regression or growth mixture uses the covariance stored with the fit and refuses any other argument with `latents_bad_argument` (an interval on another covariance is `get_results(x, "coefficients", vcov_type = )`); `vcov()` warns `latents_unused_argument` for an argument it does not take, such as `vcov_type =` in place of `type =`.

Mixture regressions (`mixture_regression()`) `"coefficients"` (one row per class and term, with Wald statistics, `p_adjusted` by Benjamini-Hochberg over non-intercepts and `exp_estimate` for the binomial, Poisson and ordinal families; ordinal thresholds lead each class as terms `"threshold:<lower>|<upper>"`), `"classes"`, `"membership"` (multinomial logits and odds ratios), `"group_classes"` (rows for two-level fits only), `"fit"` (`bic` penalised by the number of independent units, `bic_rows` by the number of rows, as flexmix reports it), `"assignments"`, `"groups"`, `"fitted"`, `"starts"` (`stage` `"screened"` or `"completed"`), `"classification"` and `"recovery"` (by = `"class"` or `"group_class"`).

Growth mixtures (`mixture_regression(random = )`) `"coefficients"`, `"classes"` (each class's share with its interval, `expected_persons`, `assigned_persons`, `average_posterior` and `residual_sd`), `"membership"`, `"fit"`, `"assignments"` (one row per person: `class`, `probability` and one `posterior_*` column per class), `"starts"` and `"recovery"`, as for a mixture regression, plus `"random"` (random-effect variances, standard deviations and correlations, with intervals on the log and Fisher-z scales), `"random_effects"` (one row per person), `"trajectories"` (each class's mean trajectory with a band) and `"individual"` (each person's predicted curve); the last two take `time`. They have no `"groups"`, `"fitted"` or `"classification"` table. A multilevel growth mixture adds `"group_classes"` and `"clusters"`, and its `"fit"` uses the number of clusters in `bic` (persons in `bic_persons`).

Additive fits (`multilpa(family = "additive")`) `"parameters"` (class means, between- and within-group variances and class weights; variance intervals on the log scale, weights on the logit scale; `adjust` sets the p-value adjustment, `"none"` by default), `"group_classes"`, `"groups"`, `"intercepts"` and `"conditional_intercepts"` (each group's posterior intercept, overall or given each class), `"assignments"`, `"classification"`, `"fit"`, `"recovery"` and `"starts"`. `effective_groups` in `"group_classes"` is `w^2 (1 - w) / Var(w)`, the number of perfectly classified groups that would estimate the class weight as precisely; below 50 the class's intervals carry a `latents_weak_class` warning.

Cross-level fits (`multilpa(family = "full_cross_level")` **or** `"restricted_cross_level"`) `"profiles"` (individual profiles, then the group classes' means of group means), `"group_classes"`, `"composition"` (profile shares within each class: model-implied for the full family, a posterior-weighted description for the restricted one), `"groups"`, `"assignments"`, `"fit"` and `"starts"`. The likelihood is a working likelihood, comparable only between cross-level fits of the same data.

General transition fits (`lta()` **with** `transitions = "occasion"`**, transition or initial covariates,** `measurement = "occas` `"transitions"` (model-implied, averaged over the observed groups, with their sampling weights in a weighted fit; `"initial"` and `"second_order_transitions"` likewise), `"transition_coefficients"` (log odds of each move against staying, with odds ratios), `"second_order_transitions"` and `"second_order_coefficients"` (`order = 2`), `"initial"` and `"initial_coefficients"`, `"profiles"`, `"responses"` (categorical indicators: `occasion`, `profile`, `indicator`, `category`, `probability`), `"ordinal"` and `"count_means"` (ordinal and count indicators), `"group_classes"`, `"assignments"`, `"fit"` and `"starts"`.

Pooled imputations (`pool_imputations()`) `"estimates"` (the pooled table), `"imputations"` (each imputation's aligned estimates) and `"fits"` (one row per imputation).

Transition and group-class family grids (`enumerate_classes(time = )`, `enumerate_classes(family = )`) `"candidates"` (one row per candidate, its name first, with its likelihood, criteria, entropy and diagnostics) or `"best"`, the row with the lowest `criterion` (`"bic"` by groups, the default, `"aic"` or `"bic_individual"`). `"best"` reports the lowest value of a criterion; it does not select a model. With one group class every family is the same model, so those rows share

a likelihood. Candidates are extracted by name with `get_candidate()`; transition grids can also be passed to `compare_models()`.

What `"all"` returns

A named list of tables included by default for the object, in catalogue order. Some results require an explicit request, including the `"lmr"` table for an enumeration object. Each included table is built with its default arguments; supplying any further argument raises `latents_bad_argument`. A table this object cannot produce (for example a sequence table for a fit made without `time`, or bivariate residuals for a family without discrete group classes) is omitted, so the names of the list show what was available. Only the conditions that mean the table does not exist for this object are skipped; any other error is raised.

Table arguments

These are passed through `...`; each table takes only those listed for it under *Tables*.

`data` A data frame with one row per observation of the fit, in the order it was fitted in. `NULL` uses the columns the fit carries. The standard errors of `"profiles"` and `"responses"` come from `parameter_inference()` on these data. Without `data`, a fit that cannot supply them (bound-active, unconverged, singular information, a noise component) gives `NA` with a `latents_no_standard_errors` message; with `data`, the underlying condition is raised.

`scale` For `"profiles"`: `"raw"` (default) reports estimates in input units; `"standardized"` divides each indicator's deviation from its grand mean by that indicator's observed standard deviation, as `plot(x, scale = "standardized")` draws. The standard errors are divided by the same constant.

`format` For the posterior tables and `"sequences"`, `"long"` (default) gives one row per unit and class and `"wide"` one row per unit with one column per class. For `"information_criteria"`, `"wide"` (default) gives one row and `"long"` one row per criterion and sample-size convention.

`level` The classification level to report: `"individuals"`, `"groups"` or `"both"`. `NULL` (default) gives `"both"` for a fit with discrete group classes and `"individuals"` otherwise, as in `diagnostics()`.

`truth` For `"assignments"`, one or more column names of `data` holding known labels to compare the assignments with. See *Recovery*.

`by` For `"residuals"`, `"profile"` (default) assesses each profile separately and `"overall"` pools them.

`adjust` For `"residuals"`, the multiplicity correction applied across the indicator pairs: `"none"` (default) or one of the `stats::p.adjust()` methods.

`definitions` For `"information_criteria"` in long format, `TRUE` adds each criterion's formula and reference. Default `FALSE`.

`estimated`, `stable` For `"transitions"`, `TRUE` or `FALSE` restricts the table to the rows estimated (or not) from observed occupancy, or to the diagonal (or off-diagonal) cells. `NULL` (default) keeps every row.

Recovery

`what = "assignments"` with `truth` returns a cross-tabulation of the model's labels against known ones instead of the per-observation table: one row per class and truth value, with columns `assignment`

(the model label compared against), class, truth (the column name), value, n and proportion (the share of units with that truth value assigned to that class). Several truth columns are stacked. Excluded observations or groups appear under class = NA, so the table distinguishes unclassified units from assignments to an estimated class.

A truth column that is constant within every group is compared against group_class; one that varies within any group is compared against profile. The assignment column records which comparison was made. Group-level recovery counts each group once. Unused factor levels are omitted. For a fit without discrete group classes every column is compared against profile.

Class labels are arbitrary, so a truth value assigned mostly to a class with a different label indicates relabelling, not misclassification.

Information criteria

Let q be the number of free parameters, n the chosen sample size, and EN the entropy of the complete-data classification (defined below). The criteria are deviance = -2L, aic = -2L + 2q, bic = -2L + q log(n), sabic = -2L + q log((n + 2) / 24), caic = -2L + q (log(n) + 1), awe = -2(L - EN) + 2q (1.5 + log(n)), icl = -2L + q log(n) + 2 EN, kic = -2L + 3(q + 1), and clc = -2L + 2 EN. clc, like icl and awe, uses the entropy sum; tidyLPA reports a CLC based on relative entropy, and the two values are not comparable. The kic form is Cavanaugh's (1999) criterion as tidyLPA computes it (Rosenberg et al. 2018), with q the number of free parameters; Cavanaugh's general form -2L + 3q is 3 smaller for every model, so differences between models and the model with the smallest value are the same under both.

EN is the soft entropy of the complete-data classification, the entropy term of the classification log likelihood log L_c = log L - EN of ICL (Biernacki, Celeux and Govaert 2000), AWE (Banfield and Raftery 1993) and CLC (Biernacki and Govaert 1997). The latent classification of a two-level model is the group class of every group and the profile of every row, so EN = sum_j H(C_j | y_j) + sum_j sum_h P(C_j = h | y_j) the group-class entropy plus each row's profile entropy within its group class. With one group class, and in every single-level model, it is the entropy of the profile posteriors. It is the soft form, the posterior expectation of -log P(classes | y) (McLachlan and Peel 2000, section 6.10.3), not the entropy of the modal (hard) assignment that Biernacki, Celeux and Govaert (2000) also discuss. A fit with several group classes saved by latents 0.97.1 or earlier lacks the rows' within-class entropies, and its awe, icl and clc are NA; refit it to obtain them.

Every criterion that depends on a sample size is reported under both multilevel conventions: "groups" with n the number of groups and "individuals" with n the number of rows. EN is the same under both, so the conventions differ only in n, and in a single-level model they coincide. Use one convention consistently across compared candidates. deviance, aic and kic carry convention = NA_character_; clc, which uses no sample size, is the same in both rows. A transition model (lta()) keeps one entropy per level instead: its profiles form a Markov chain within a person, whose path entropy is not the sum of the occasions' entropies. Its "groups" criteria use group-class entropy, whereas its "individuals" criteria use occasion-level profile entropy. These definitions are reported by get_results(x, "information_criteria", format = "long", definitions = TRUE). Individual sample sizes exclude rows with no observed indicators, and group sample sizes exclude groups in which every row has none (n_informative_groups); the entropy and the classification statistics exclude the same rows and groups, as in Mplus. Units of sampling weight zero are in no sample size and no entropy, under either drop_empty. For a fit with sampling weights, L is the pseudo log likelihood and EN the sampling-weighted entropy, each unit's terms multiplied by its weight, so that integer weights give the entropy of the data with each unit repeated that many times.

The "entropy" table, relative_entropy, the average posteriors, the odds of correct classification, the classification errors and the BCH weights are weighted the same way; modal counts stay counts.

Classification quality

The odds of correct classification for class k is (p / (1 - p)) / (r / (1 - r)), where p is the average posterior in the assigned class and r is the model-estimated class proportion. Values near one indicate classification no better than the class proportions alone. A value of five is a conventional descriptive benchmark, rather than a formal test of class separation. It is NA_real_ when average_posterior is missing or zero, or when estimated_proportion is zero or one. A perfect average_posterior of one gives Inf at an interior estimated proportion; a single-class solution always reports NA_real_. average_posterior is NA_real_ for a class with no modal members. n_modal and estimated_n use modal assignments and posterior probabilities, respectively, and need not coincide.

relative_entropy is 1 - EN_level / (n log K) (Ramaswamy et al. 1993), between zero and one, higher meaning sharper classification, with EN_level the level's own entropy sum, reported as entropy_sum, lower meaning sharper. With one group class the individual level's entropy_sum is the EN of the information criteria; with several, that EN adds the group-class entropy to the profile entropy within each group class (see *Information criteria*). relative_entropy is NA_real_ for a level with a single class.

A fit with individual profiles but no discrete group classes reports individuals only for level = "both" and raises latents_no_group_classes for level = "groups".

Bivariate residuals

"residuals" checks, pair by pair, the assumption that indicators are independent within a profile, weighting each observation by its posterior (times its sampling weight). Every pair of indicators has a row (per profile under by = "profile", once under by = "overall"). data must be in the order the model was fitted in; the alignment is checked. Large residuals suggest associations that the measurement model does not adequately represent. Calculations for each kind are described below:

- "gaussian", two Gaussian indicators: observed is the posterior-weighted correlation of the residuals from the profile's means (pooled over the profiles under "overall"), expected the correlation the fitted covariance implies (zero for a diagonal one). statistic is the difference of their Fisher z transforms times sqrt(effective_n - 3), referred to the standard normal (Fisher, 1921), with a two-sided p-value; df is NA. This is an approximate reference distribution for a posterior-weighted diagnostic.
- "categorical", two categorical or ordinal indicators (an ordinal indicator's categories are its levels): statistic is the Pearson X^2 of the posterior-weighted two-way table against the table the model implies, which is Latent GOLD's bivariate residual times its df, and p_value refers it to chi-square with df = (R - 1)(C - 1). observed and expected are the two tables' Cramer's V (Cramer, 1946), zero for the model's table within a profile. The chi-square reference ignores that the model's table is estimated from the same data and can therefore be conservative (Oberski, van Kollenburg & Vermunt, 2013). Interpret p_value together with residual and statistic; a non-significant result does not establish conditional independence.

- `"mixed"` (a Gaussian and a categorical or ordinal indicator) and `"count"` (any pair with a count indicator): `observed` is the posterior-weighted correlation of the two indicators' values within the profile, centred on the profile's weighted means (pooled over the profiles under `"overall"`); a categorical or ordinal indicator enters as its category number in category order, which for a nominal indicator with more than two categories depends on that order. `expected` is zero, the correlation of independent indicators, and `statistic`, `df` and `p_value` are `NA`: no reference distribution is computed for these pairs.

A half-observed row of a joined `local_dependence` pair enters that pair's tables only where both items are observed. `p_adjusted` applies the correction named by `adjust` across the pairs that have a p-value. A fit with sampling weights uses them in the descriptive associations and `effective_n`; design-based inference is not implemented, so `statistic`, `df`, `p_value` and `p_adjusted` are `NA` for weighted fits.

References

Fisher, R. A. (1921). On the "probable error" of a coefficient of correlation deduced from a small sample. *Metron*, 1, 3–32.

Cramer, H. (1946). *Mathematical Methods of Statistics*. Princeton University Press.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6, 461–464.

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

Bozdogan, H. (1987). Model selection and Akaike's information criterion. *Psychometrika*, 52, 345–370.

Banfield, J. D., & Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49, 803–821.

Biernacki, C., & Govaert, G. (1997). Using the classification likelihood to choose the number of clusters. *Computing Science and Statistics*, 29, 451–457.

Cavanaugh, J. E. (1999). A large-sample model selection criterion based on Kullback's symmetric divergence. *Statistics and Probability Letters*, 42, 333–343.

Rosenberg, J. M., Beymer, P. N., Anderson, D. J., Van Lissa, C. J., & Schmidt, J. A. (2018). tidyLPA: An R package to easily carry out latent profile analysis (LPA) using open-source or commercial software. *Journal of Open Source Software*, 3(30), 978.

McLachlan, G., & Peel, D. (2000). *Finite Mixture Models*. Wiley.

Ramaswamy, V., DeSarbo, W. S., Reibstein, D. J., & Robinson, W. T. (1993). An empirical pooling approach for estimating marketing mix elasticities with PIMS data. *Marketing Science*, 12, 103–124.

Biernacki, C., Celeux, G., & Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 719–725.

Bolck, A., Croon, M., & Hagenaars, J. (2004). Estimating latent structure models with categorical variables. *Political Analysis*, 12, 3–27.

Nagin, D. S. (2005). *Group-Based Modeling of Development*. Harvard University Press.

Vermunt, J. K. (2010). Latent class modeling with covariates: two improved three-step approaches. *Political Analysis*, 18, 450–469.

Satopaa, V., Albrecht, J., Irwin, D., & Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. *Proceedings of the 31st International Conference on Distributed Computing Systems Workshops*, 166–171.

Oberski, D. L., van Kollenburg, G. H., & Vermunt, J. K. (2013). A Monte Carlo evaluation of three methods to detect local dependence in binary data latent class models. *Advances in Data Analysis and Classification*, 7, 267–279.

See Also

`as.data.frame()`, which returns the primary table alone, `summary()`, which prints every table, and `diagnostics()` for the classification tables with a diagnostic summary.

Examples

```
fit <- multilpa(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student", n_profiles = 2, n_group_classes = 2, n_starts = 4,
  seed = 1
)
get_results(fit)
get_results(fit, what = "profile_probabilities")
get_results(fit, what = "entropy")
names(get_results(fit, what = "all"))

# Compare estimated assignments with the generating classes at both levels.
get_results(fit, what = "assignments", data = course_engagement,
         truth = c("engagement", "student_type"))
```

---

get_tna *Represent transitions between latent states as a network*

Description

`get_tna()` constructs a network summarising transitions between latent profiles or classes across ordered observations. The profiles or classes form the nodes, and directed edge weights represent transition probabilities, including the probability of remaining in the same state. The returned network supports visualisation and analysis with the tna package.

For a supported latent transition model, the network uses model-estimated transitions, pooling expected transitions across latent group classes. For a profile or class model fitted with `time`, it instead summarises transitions between the most probable class assignments in each group's ordered sequence.

Usage

```
get_tna(x, ...)
```

Arguments

x	A time-homogeneous fitted model from `lta()`, or a `multilpa()` or `multilca()` model fitted with `time`, for which the network is based on modal profile sequences.
...	Passed to `tna::tna()`.

Details

For a supported latent transition model, transition probabilities are the expected transition counts summed across group classes and normalised by row, so each class contributes in proportion to its share of the expected transitions, not its class probability. If a profile has no expected outgoing transitions in any class, its row is the class rows averaged by class probability, and a warning states that the row is unestimated; absent transitions are not treated as self-transitions.

For a `multilpa()` or `multilca()` fit, `tna::tna()` builds the network from the modal sequences. A fit made without `time` is refused.

A general `lta()` fit (occasion-varying or covariate-dependent transitions, second order, movers and stayers, or ordinal, count or weighted data) is currently unsupported and raises `latents_unsupported_tna`; its transitions are returned by `get_results(x, "transitions")`.

Value

An object of class `tna`, as returned by `tna::tna()`, to which the functions of the tna package apply, including `centralities()`, `communities()`, `cliques()` and `plot()`.

Interpretation

A network built from modal assignments is a descriptive summary and does not account for classification uncertainty. A pooled network may conceal differences between latent group classes; use `get_group_tna()` to examine their separate networks.

See Also

`get_group_tna()` for one network per latent group class.

Examples

```
if (requireNamespace("tna", quietly = TRUE)) {
  activity <- c("browse", "lectures", "forum_read")
  moves <- lta(course_engagement, vars = activity, id = "student",
                           time = "sequence", n_profiles = 2,
                           n_group_classes = 2, n_starts = 2, seed = 1)
  get_tna(moves)
}
```

`growth_schools` *Reading growth of students nested in schools*

Description

`growth_schools` is a simulated dataset illustrating multilevel growth mixture models. It contains repeated reading scores for 600 students nested within 40 schools, with up to five observations per student.

The data were generated from two student trajectory classes, labelled "improving" and "plateauing". Students vary in their initial reading achievement and rate of change within each class. Two school-level classes differ in the prevalence of these student trajectories.

The dataset illustrates how multilevel growth mixture models represent individual variation in development alongside differences in trajectory membership between schools. Generating class labels are retained at both levels for assessing recovery.

Usage

```
growth_schools
```

Format

A data frame with 2780 rows (600 students, 40 schools) and 7 columns:

school Integer school identifier, 1 to 40. Pass it as `cluster`.

student Integer student identifier, 1 to 600. Pass it as `id`.

wave Integer test occasion, 0 to 4. The time variable.

score Reading score.

programme Standardized school support programme, constant within school.

school_type Factor, `"supportive"` or `"struggling"`: the school's generating class; retained for assessing class recovery.

trajectory Factor, `"improving"` or `"plateauing"`: the student's generating class; retained for assessing class recovery.

Details

There are 15 students per school, with occasions indexed by `wave` from 0 to 4. The generating student classes have two linear mean trajectories: *improving* (intercept 48, slope 3 points per occasion) and *plateauing* (intercept 56, slope 0.3). Random intercepts and slopes have standard deviations 3 and 0.6 and correlation -0.2; the residual standard deviation is 2. The generating school classes, labelled *supportive* and *struggling*, have improving-class probabilities of 0.8 and 0.25, respectively. The standardized school-level `programme` variable predicts school-class membership. Approximately 8% of test occasions were removed at random.

The school types correspond to the group classes of `mixture_regression(..., cluster = "school", n_group_classes = 2)`, with `group_membership = "programme"`.

Source

Simulated by `data-raw/growth-schools.R` in the source repository.

See Also

`mixture_regression()`, `get_results()`.

Examples

```
descriptives(growth_schools, c("score", "programme"))

fit <- mixture_regression(score ~ wave, growth_schools, n_classes = 2,
                          id = "student", class_level = "group",
                          random = "wave", random_covariance = "equal",
                          cluster = "school", n_group_classes = 2,
                          group_membership = "programme",
                          n_starts = 2, seed = 1)
get_results(fit, "group_classes")
get_results(fit, "recovery", data = growth_schools, truth = "trajectory")
```

---

growth_scores *Simulated achievement scores with three growth trajectories*

Description

`growth_scores` is a simulated longitudinal dataset illustrating growth mixture models. Each row represents an achievement score for one of 300 students, measured on up to seven occasions.

The data were generated from three trajectory classes, labelled "improving", "stable" and "declining". Within each class, students differ in their initial achievement and rate of change through random intercepts and slopes.

The dataset illustrates how growth mixture models distinguish differences between class mean trajectories from variation among individuals within the same class. Generating class labels are retained for assessing recovery, and randomly omitted observations produce an unbalanced panel.

Usage

```
growth_scores
```

Format

A data frame with 1844 rows (300 students) and 5 columns:

student Integer student identifier, 1 to 300. Pass it as `id`.

wave Integer test occasion, 0 to 6. The time variable.

score Achievement score.

motivation Standardized motivation, constant within student.

trajectory Factor, `"improving"`, `"stable"` or `"declining"`: the student's generating class; retained for assessing class recovery.

Details

Occasions are indexed by `wave`, from 0 to 6. The generating classes have three linear mean trajectories: *improving* (intercept 48, slope 3.5 points per occasion), *stable* (intercept 62, slope 0.5), and *declining* (intercept 58, slope -2.5). Within each class, individual trajectories vary according to random intercepts and slopes with standard deviations 4 and 0.8 and correlation -0.3. The residual standard deviation is 3. Motivation is a time-invariant predictor of class membership, with higher values associated with greater probability of the improving class and lower probability of the declining class. Approximately 12% of test occasions were removed at random, producing an unbalanced panel.

A model without random effects may represent within-class variation by additional trajectory classes.

Source

Simulated by `data-raw/growth-scores.R` in the source repository.

See Also

`mixture_regression()`, `compare_models()`.

Examples

```
descriptives(growth_scores, c("score", "motivation"))

fit <- mixture_regression(score ~ wave, growth_scores, n_classes = 3,
                          id = "student", class_level = "group",
                          random = "wave", random_covariance = "equal",
                          n_starts = 4, seed = 1)
get_results(fit, "recovery", data = growth_scores, truth = "trajectory")
```

`latents-conditions` *Errors, warnings and diagnostic messages*

Description

latents reports errors, warnings and messages to communicate problems with input data, difficulties during estimation, and limitations of requested analyses. Errors indicate that an operation could not be completed. Warnings identify issues requiring attention when interpreting or using a result. Messages provide progress updates or supplementary information.

Details

This reference describes the conditions that trigger these signals and their associated classes. In automated analyses, conditions can be handled by class using `tryCatch()` or `withCallingHandlers()`. Class-based handling allows scripts to respond to specific situations without depending on message wording, which may change between versions.

Value

Nothing; this page documents conditions rather than a function.

Input validation

`latents_bad_data` The data frame, the indicator columns or the group column are invalid: too few rows, a duplicated or absent column name, an indicator that is not a numeric vector, an indicator that is constant or entirely unobserved, missing or non-finite values under `missing = "error"`, or a missing group identifier.

`latents_unweightable_level` The stored sampling weights cannot be aligned with a classification level, for example in an older cross-level fit without a row-to-group mapping. `diagnostics()` handles this condition internally and reports the affected checks as `n/a`.

`latents_not_a_fit` A function requiring a fitted model received an unsupported object or a model from an unsupported family. The message identifies the supplied object. Also inherits from `latents_bad_argument`.

`latents_bad_argument` An argument has an invalid value: a scalar such as `n_profiles`, `n_group_classes`, `n_starts`, `max_iter`, `tol` or `min_variance`; a `seed` or `seeds` that is not a whole number accepted by `set.seed()`; a `what` naming a table or view the object does not have, or a `what` that is `NA`, not text or more than one name; an enumerated option such as `missing`, `boundary`, `select_start`, `acceleration`, `centering`, `scaling`, `variance_model`, `covariance_model`, `vcov_type` or `adjust` given a value that is not one of its choices; or an argument the function or table does not use.

`latents_bad_time` `time` is missing, or repeats a position within a group.

`latents_bad_categorical` A categorical indicator cannot be encoded, or `min_probability` is too large for its number of categories.

`latents_bad_start` Supplied starting values lack a block the model needs, have the wrong shape, or carry labels naming an indicator or category the fit does not have.

`latents_bad_fixed` A `fixed` request names a block the model does not have or that `start` does not supply, or `start` is absent.

`latents_bad_stage` A `measurement` fit given to `fit_staged()` already has more than one group class; was fitted to different indicators, profiles or measurement options than the stage requests; or was fitted with `weights`, `centering`, or ordinal or count indicators.

`latents_bad_transition` A transition model was requested with fewer than two profiles, or with no group observed at two occasions.

`latents_bad_inference_data` Data supplied with a fitted model differ from the fitting data in row count, values or group ordering.

`latents_no_indicator_data` A fit carries no stored indicators, so the request needs the original data.

`latents_incomplete_fit` A fit lacks fields the request needs, usually because it was made by an older version.

Estimation and identification

`latents_unidentified` The requested model is not identified from the data: more profiles than distinct observed indicator rows, more group classes than groups, or more than one group class

with only one profile or only singleton groups. Raised as a warning, and the fit returned, for a model of complete categorical and ordinal indicators with one group class that has more free parameters than free cells of its table (negative degrees of freedom); a model with membership covariates is not judged by that count.

`latents_all_starts_failed` Every EM start failed, so no fit was produced. The message gives the distinct start errors.

`latents_empty_profile` A profile has no observed responses for an indicator, so its response probabilities are not identified; or a group class has no effective membership, so its profile prevalences are not defined.

`latents_too_few_groups` There are too few independent groups to form the covariance of the reported quantities. Raised by robust and OPG `parameter_inference()`, and by `three_step()` and `r3step()`, whose cluster-robust contributions sum to zero at the estimate and so span at most one dimension fewer than the number of groups.

`latents_inseparable_classes` The classes are not separated well enough for the requested three-step correction.

`latents_too_many_cores` `cores` exceeds the machine's core count; the fit uses the machine's count instead.

`latents_no_converge` A fit required by the request did not converge.

`latents_vanished_density` Every component density is zero, or one overflowed, at a point EM reached: the start is recorded as failed (`get_results(x, "starts")`), and a fit whose every start fails raises `latents_all_starts_failed`.

`latents_boundary_fit` A variance or response probability is at its bound, a mixing probability is zero, or a membership logit coefficient diverges (separation). The usual normal approximation for Wald inference does not apply at the parameter boundary, so standard errors are unavailable under the requested inference method.

`latents_singular_information` The observed information is not positive definite or is numerically singular, so no standard error can be formed.

`latents_unknown_candidate` The requested profile and group-class combination was not enumerated.

`latents_failed_candidate` An enumerated candidate could not be fitted. The message gives that candidate's error.

`latents_no_group_classes` A table, level or view that needs discrete group classes was requested from a model without them, for example `level = "groups"` for a fit without group classes, or a group-level plot of a fit with fewer than two group classes.

Model comparison

`latents_bad_nesting` Two models are not nested as the comparison requires, including a `bootstrap_lrt()` pair whose held measurement blocks differ, are held at different values, or are held in only one of the two models.

`latents_incomparable_models` Two models were fitted to different data, or differ in covariance structure, centering, scaling mode or `drop_empty` where the comparison requires them to match.

`latents_parallel_failure` The worker cluster could not be started, or a broken cluster's surviving processes could not be stopped safely. A process lost during fitting is otherwise retried and reported with `latents_worker_lost` instead.

`latents_worker_lost` One or more worker tasks returned no result, possibly because the operating system terminated them for excessive memory use. Missing results are retried once with fewer workers. The warning reports the numbers lost, recovered and still failed, and suggests fewer `cores`. Persistent losses become failed starts or candidate rows, retaining other results. After a socket connection failure, the affected batch is retried and results from previously completed batches are retained.

`latents_suspect_categorical` A numeric indicator fitted as categorical has non-integer values or more than ten categories, so it may be continuous. Every distinct value is fitted as a category. The warning names the indicators (field `indicators`) and is given once per call, including an enumeration.

`latents_progress` A message from `enumerate_classes()` with `verbose = TRUE` (the default when `cores > 1`): one as the candidates are prepared, one as their starts are fitted, and one when the grid is done. Muffle it with `suppressMessages(classes = "latents_progress")` or `verbose = FALSE`.

`latents_candidate_failed` A candidate of an enumeration that was fitted as one worker task lost that worker twice. It is recorded as the candidate's error, in its row of the grid, and the other candidates are returned.

`latents_reversed_likelihood` The larger nested model has a lower likelihood, indicating that at least one fit has not reached its global maximum.

`latents_bad_outcome` A distal outcome given to `three_step()` is of a type the requested method cannot handle, varies within a group when a group-level outcome is requested, or is a measurement indicator.

`latents_bad_covariate` A predictor given to `r3step()` varies within a group when group-level membership is predicted, makes the design rank deficient, or is a measurement indicator; or a predictor given to `multilpa(profile_covariates =, group_covariates =)` is an indicator or the group identifier, or makes the membership design rank deficient.

`multilpa_indicator_reused` A measurement indicator was given as an outcome to `three_step()` or as a predictor to `r3step()`. Raised together with `latents_bad_outcome` or `latents_bad_covariate`.

`latents_bad_scores` Score contributions for the robust sandwich could not be formed.

Unsupported requests

These are raised when the package does not implement the requested analysis.

`latents_unsupported_inference` Standard errors are not available for this fit. Raised by `parameter_inference()` for a fit made with `prior` (Wald method), for the cross-level families, for a covariate fit made by an older version, and for a fit recording a covariance structure code this version does not know; and by `parameter_inference(method = "bootstrap")` for an additive or covariate fit, a multilevel growth mixture, or a fit holding a measurement block through `fixed`. Also raised by `vcov(scale = "unconstrained")` for a bootstrap covariance.

`latents_unsupported_prior` `multilpa(prior = )` was combined with an option the conjugate prior is not defined for: the VEE, EVE, VVE or EVV covariance structure (mclust defines none), categorical indicators, a `fixed` block, membership covariates, or missing indicator values.

`latents_unsupported_noise` `multilpa(noise = TRUE)` was combined with more than one group class, categorical indicators, a `fixed` block, membership covariates or missing indicator values; or a fit with a noise component was given to a function that does not support one: `parameter_inference()`, `three_step()`, `r3step()`, `bootstrap_lrt()`, `starting_values()`, bivariate residuals, `fit_staged()` or a posterior plot. Also raised by `predict()` for partially observed rows when the fitted model has a noise component, because its stored hypervolume does not define the required marginal noise density.

`latents_unsupported_structure` `multilpa()` was asked for a covariance structure other than EEI, VVI, EEE or VVV together with `profile_covariates` or `group_covariates`.

`latents_bad_weights` `weights` does not name a column, or its values are missing, negative, non-finite, vary within a unit, or are positive for fewer than two units.

`latents_unsupported_weights` A weighted fit was given to a function or option that does not support sampling weights: `bootstrap_lrt()`, `enumerate_regressions(bootstrap = )`, `three_step()`, `r3step()`, `prior` or `noise`; or a `vcov_type` other than `"robust"` was requested for it.

`latents_unsupported_indicator` `ordinal` or `count` indicators were combined with an option that does not support them: membership covariates, `start`, `fixed`, `prior` or `noise`.

`latents_pooling_failed` A fit or its inference failed for one imputation in `pool_imputations()`, or the imputations' fits report different parameters. No imputation is dropped; the message names the imputation and gives the original reason.

`latents_bootstrap_failed` Fewer than two resamples produced a usable fit. Raised by `parameter_inference(method = "bootstrap")`; the message gives the first reason a resample failed.

`latents_no_free_parameters` The fit holds every parameter fixed, so there is no standard error to report. Raised by `parameter_inference()` and `vcov()`.

`latents_held_parameter` A parameter named in `parm` was held fixed by the fit and has no confidence interval. Raised by `confint()`.

`latents_unsupported_sensitivity` `sensitivity()` cannot refit the requested model or align its class labels. This includes directly fitted fixed-measurement models whose original free starting values were not retained.

`latents_unsupported_three_step` `three_step()` or `r3step()` was called on a fit the correction does not cover: a membership-covariate fit, whose classification errors depend on the predictors (the correction uses one unconditional error matrix); a multilevel growth mixture; a general `lta()` fit (occasion-varying, second-order or covariate transitions, occasion-specific measurement, a stayer class, or ordinal or count indicators), except that `r3step()` takes `initial_covariates` and `transition_covariates` for a covariate-free one; an additive, dispersion or cross-level `multilpa()` fit; or a mixture regression with observation classes.

`latents_unsupported_bootstrap` `bootstrap_lrt()` was called on a person-centred fit, for which raw data cannot be simulated under the null model because the group baselines were removed before estimation.

`latents_unsupported_tna` `get_tna()` or `get_group_tna()` was called on a general transition fit (an `lta()` fit with occasion-varying or covariate-dependent transitions, a second order, movers and stayers, or ordinal, count or weighted data), which has no single transition matrix; its transitions are given by `get_results(x, "transitions")`.

`latents_no_plot` `plot(x, what = "all")` was called on an object whose plot method has no views to draw.

`latents_no_time` A sequence table or function was requested for a fit made without `time`.

`latents_unsupported_prediction` `predict()` was called on a growth mixture, a membership-covariate fit or a latent transition fit, for which prediction on new rows is not implemented; results for the fitted rows are available from `get_results()`.

`latents_missing_package` A required package from `Suggests` is not installed. Raised by `get_tna()` and `get_group_tna()` when `tna` is absent; the message names the package and how to install it.

Plotting

`latents_no_continuous`, `latents_no_categorical` The requested panel needs indicators of a kind the fit does not have.

`latents_nothing_to_describe` `descriptives()` has no variables to describe: `vars` named none, or the frame has no numeric column outside `id` and `by`.

`latents_nothing_to_plot` The requested panel has no rows to draw.

`latents_inaccurate_probability` Numerical integration for a Lo-Mendell-Rubin p-value did not meet the required accuracy, so the p-value is unavailable.

`latents_no_local_dependence` `get_results(fit, "local_dependence")` on a fit made without `local_dependence`.

`latents_no_direct_effects` `get_results(fit, "direct_effects")` on a fit made without `direct_effects`.

`latents_partial_pair` Not raised. A row observing one item of a `local_dependence` pair contributes that item, with the pair's other item summed out.

`latents_no_lmr` `get_results(x, "lmr")` on a grid with no two converged candidates one profile or one group class apart. Also of class `latents_nothing_to_plot`.

`latents_invalid_lmr` The alternative of a nested LMR comparison has a lower likelihood than the null beyond numerical roundoff. The comparison table retains its row with unavailable statistics and the reason in `note`; improve the alternative's fit before comparing.

`latents_no_tree` A tree would have fewer than two rows, or the group level of a grid with one group class was asked for. Also of class `latents_nothing_to_plot`; `get_results(x, "all")` leaves the `"tree"` table out.

`latents_unsupported_view` A grid function (`plot_enumeration()`, `plot_selection()`, `plot_tree()` or `plot()` of a grid) was asked for a view that this kind of grid cannot draw, such as the tree of a group-class family or regression grid. The message gives the reason and the views the grid offers; `plot_views(x)` lists them. Also of class `latents_bad_argument`.

`latents_unknown_category`, `latents_unknown_criterion` A named category or information criterion does not exist in the fit. An unknown criterion of a grid is also of class `latents_bad_argument`.

`latents_bad_scale` An unrecognised plotting scale was requested, or `scale = "standardized"` met an indicator with zero or undefined standard deviation.

Additional errors, warnings and messages

These conditions report limitations of a returned result or an inability to complete estimation. Message-only conditions are identified explicitly.

`latents_failed_starts` Some, but not all, EM starts failed. The remaining starts are in `get_results(fit, "starts")`.

`latents_not_identified` `mixture_regression()` with a binary outcome and one trial per class assignment, where a mixture of Bernoulli regressions is not identified.

`latents_missing_data` `mixture_regression()` with `missing = "error"` met a missing value in a variable the model uses.

`latents_rows_dropped` `mixture_regression()` with `missing = "omit"` dropped incomplete rows; the message states how many.

`latents_no_valid_start` Every `mixture_regression()` start degenerated: a class emptied or its variance collapsed.

`latents_degenerate_start` Some `mixture_regression()` starts degenerated and were set aside.

`latents_separation` A `mixture_regression()` logit or log-link coefficient diverged (perfect separation); it and its standard error are unreliable.

`latents_em_decrease` The `mixture_regression()` EM likelihood decreased beyond rounding error. This indicates a defect and should be reported.

`latents_unconverged` The selected start did not satisfy the convergence criterion before `max_iter`, or an inference check found a non-negligible score. The returned estimate requires further assessment.

`latents_boundary` A variance, or a covariance eigenvalue, is at `min_variance`, so the fit is boundary-active and its standard errors are not valid at that parameter.

`latents_atomic_groups` Under `centering = "person"` with `scaling = "person"`, some groups have fewer than three observations of an indicator. Standardizing two values within a group makes them exactly `-1/sqrt(2)` and `1/sqrt(2)`, so those rows carry only a sign pattern, and a profile can collapse onto them. The message gives the share of rows in such groups. When every group is like that, the fit is refused with `latents_bad_data` instead.

`latents_small_classes` A profile or group class has an effective membership below one observation; or, in `three_step()` with any `method`, a class is carried by a single unit (fewer than 1.5 effective carrying units, by Kish's formula), so its standard error is reported as `NA`.

`latents_thin_profiles` A Gaussian component with free variances of its own has posterior membership mass below the number of free parameters in its Gaussian density. This is a heuristic prompt to inspect the solution, not a proof of singularity or nonidentification. Covariances held at zero and held means are excluded, and held variances do not trigger it. Memberships use normalized sampling weights and only rows with an observed Gaussian indicator; this mass is not an independent sample count or a weight effective sample size. Transition models check occasion-specific measurement separately; cross-level models check both Gaussian levels. The estimate is returned unchanged. See `get_results(fit, "gaussian_diagnostics")` for continuous covariance and support diagnostics; none supplies an automatic spike classification or an identification guarantee.

`latents_all_starts_boundary` `boundary = "exclude"` found every start with a variance at `min_variance`, so none was set aside and the best start across all of them is reported.

`latents_weak_class` `multilpa(family = "additive")` only: a group class is supported by fewer than 50 effective groups (see `effective_groups` in `get_results()`), because it is small, poorly separated, or both. Standard errors are reported, but intervals for class means and weights may be miscalibrated.

`latents_failed_replicates` Some bootstrap replicates failed validation, so the bootstrap p-value is `NA`.

`latents_single_level` A message, not a warning: `id = NULL` was passed, so the fit has one observation per unit and no second level. Raised by `multilpa()` on every such fit and once by an `enumerate_classes()` grid; `suppressMessages()` or a `latents_single_level` handler silences it.

`latents_bootstrap_dropped` Some resamples did not produce a usable fit and were excluded from the interval. Raised by `parameter_inference(method = "bootstrap")`; the message states how many.

`latents_sensitivity_dropped` One or more seeds did not produce a fit in `sensitivity()`. Their rows are `NA`, and the warning states how many failed and the first reason.

`latents_extreme_coefficients` A membership logit coefficient is extreme after adjustment for covariate location and scale. This may indicate separation or sparse class membership and requires inspection before interpreting the coefficient or its standard error.

`latents_empty_transition_row` A profile is never occupied before a final occasion, so its transition row is uniform by construction rather than estimated; `get_results(fit, "transitions", estimated = FALSE)` lists such rows. The general `lta()` engine raises it too, for an origin profile (or pair, under `order = 2`) that no move leaves: its logits are fixed at zero and marked `estimated = FALSE` in `get_results(fit, "transitions")` and `parameter_inference()`. In both engines such a row is not counted in `n_parameters`, AIC or BIC. The transition-network handoff also warns when it includes such a row.

`latents_unidentified_occasion` An `lta()` fit with `transitions = "occasion"` has an occasion at which no row observes an indicator, followed by an occasion that does. The moves into and out of the empty occasion (or, for an empty first occasion, the initial distribution and the first move) enter the likelihood only through their product, so their logits are not separately identified: the distinct parameter combinations yield the same likelihood. Their standard errors are not interpretable, although the parameter count includes them. Occasions after the last observed one are not counted and do not warn.

`latents_unverified_alignment` A supplied `data` frame has too little fitted information to verify row order (for example only a repeated group identifier). The result assumes the rows are in fitting order.

`latents_random_boundary` A growth mixture's random-effect covariance is degenerate in a class (a variance at its floor, or perfectly correlated random effects), or, with `residual_correlation`, the residual variance, process variance or process rate is on its bound; the standard errors of those parameters are not reported.

`latents_singular_covariance` A growth mixture with correlated residuals met a marginal covariance that is numerically singular (a process variance far larger than the residual variance). During fitting this is treated as a rejected step; it reaches the caller only for parameter values supplied explicitly.

`latents_unused_argument` A method that accepts `...` (several `plot()` methods, `summary()` of the fits and grids whose summary takes no `level` or `vcov_type`, `vcov()`, `predict()` and `descriptives()` of a data frame) was given an argument it does not use. The call proceeds and the warning names the argument.

`latents_unused_category` An ordinal indicator is a factor with levels never observed in the data. Their intercepts diverge, so Wald inference is refused; call `droplevels()` before fitting. It is a warning rather than an error because bootstrap refits keep the fitted categories.

`latents_no_standard_errors` A message, not a warning: a table that includes standard errors by default could not compute them for this fit (a bound-active or unconverged fit, singular information, or a family or noise component without inference), so they are `NA` and the message gives the reason. Passing `data` to `get_results(x, "profiles")` or `"responses"` raises the underlying condition instead.

Examples

```
set.seed(1)
example_data <- data.frame(
  school = rep(seq_len(8), each = 6),
  score_a = stats::rnorm(48), score_b = stats::rnorm(48)
)
fit <- multilpa(example_data, c("score_a", "score_b"), "school",
                n_profiles = 2, n_group_classes = 1, n_starts = 2, seed = 1)
# Handle a missing-time condition by its class.
tryCatch(get_results(fit, "sequences"),
         latents_no_time = function(condition) {
  "No occasion ordering was specified for this fit."
})
```

---

latents-plots *Plot model estimates, classification summaries, and comparisons*

Description

The plotting functions support interpretation and comparison of fitted latent variable models. Depending on the model, they display differences between classes in indicator patterns, outcome relationships, prevalence or transition probabilities. Classification plots describe how strongly observations are associated with their assigned classes.

For candidate models, plots compare information criteria and classification quality across alternative solutions. They also display convergence diagnostics and correspondence between classes, helping assess how the solution changes with the number of classes or the model specification.

Available views depend on the fitted model. Plots are returned as ggplot objects that can be customised and saved for reporting.

Usage

```
plot_profiles(
  x,
  statistic = c("mean", "median", "mode"),
  scale = c("raw", "standardized"),
  intervals = TRUE,
  labels = TRUE,
  data = NULL,
  main = NULL,
  subtitle = NULL,
```

```
  what = c("lines", "bars", "heatmap", "raincloud", "parallel", "pairs", "all")
)

plot_responses(
  x,
  category = "last",
  intervals = TRUE,
  labels = TRUE,
  data = NULL,
  main = NULL,
  subtitle = NULL
)

plot_sequences(
  x,
  labels = TRUE,
  cell_labels = TRUE,
  main = NULL,
  subtitle = NULL
)

plot_prevalence(x, labels = TRUE, main = NULL, subtitle = NULL)

plot_diagnostics(
  x,
  what = c("all", "sizes", "posteriors", "entropy", "avepp", "group_sizes",
    "group_posteriors", "group_entropy", "group_avepp"),
  main = NULL,
  subtitle = NULL
)

plot_transitions(x, main = NULL, subtitle = NULL)

plot_tree(
  x,
  main = NULL,
  subtitle = NULL,
  n_group_classes = NULL,
  n_profiles = NULL,
  model = NULL,
  candidates = NULL,
  from = NULL,
  to = NULL,
  level = c("individuals", "groups"),
  combine = FALSE
)

plot_selection(
```

```
  x,
  what = c("all", "deltas", "heatmap", "smallest", "trust", "diagnostics"),
  using = "bic_groups",
  cutoff = 0.8,
  main = NULL,
  subtitle = NULL
)

plot_enumeration(
  x,
  what = c("aic", "bic_groups", "bic_individual", "icl_individual"),
  combine = TRUE,
  labels = TRUE,
  mark_minimum = TRUE,
  main = NULL,
  subtitle = NULL,
  cutoff = 0.8,
  using = "bic_groups",
  ...
)

plot_views(x = NULL)
```

Arguments

x	The object to plot. For `plot_profiles()`, `plot_responses()`, `plot_sequences()`, `plot_prevalence()`, `plot_diagnostics()` and `plot_transitions()`, any fitted model of the package: profile, class and transition models from `multilpa()`, `multilca()`, `lpa()`, `lca()` and `lta()` (with or without covariates) take the views described under `what` and *Views by object class*; growth mixtures and mixture regressions from `mixture_regression()`, additive, dispersion and cross-level fits from `multilpa()`, pooled imputations from `pool_imputations()` and general `lta()` fits (occasion-varying transitions, covariates, mover-stayer, second order, weights, ordinal or count indicators) take the views listed under *Named plot functions on other fitted models*. For `plot_enumeration()`, `plot_selection()` and `plot_tree()`, a grid from `enumerate_classes()`, `enumerate_lpa()`, `enumerate_lca()` or `enumerate_regressions()`; each refuses the views that grid does not offer (see above). Optionally a grid for `plot_views()`.
statistic	`"mean"` (default), `"median"` or `"mode"`. They coincide for continuous indicators; for categorical ones they summarise the fitted probabilities on the numeric category scale, without intervals.
scale	`"raw"` (default) keeps each indicator in its own units; `"standardized"` divides its deviation from the observed mean by its observed standard deviation.
intervals	For profile, bar and response views, `TRUE` (default) draws 95% intervals when the fit has standard errors; otherwise the subtitle states why there are none. They are the `conf_low` and `conf_high` of `parameter_inference()`; a response probability's interval is formed on the logit scale.

`labels` — `TRUE` (default) labels each series directly; `FALSE` uses a legend. On sequence views, `TRUE` groups the rows by group class under labelled strips.

`data` — The data the model was fitted to, used only to compute the intervals. `NULL` (default) uses the data stored in the fit.

`main`, `subtitle` — Title and subtitle. `NULL` (default) uses the view's default title and subtitle.

`what` — The view to draw. `"all"` draws every view of that function that the model supports, as a list, and names in a message any it cannot draw. On fitted models other than profile, class and transition models, `plot_profiles()` draws only `"lines"` and `plot_diagnostics()` only `"posteriors"`, where that model has the view (see *Named plot functions on other fitted models*). For `plot_profiles()`: `"lines"` (default), `"bars"` (means from zero with 95% intervals), `"heatmap"` (means in observed standard deviations, or the response probabilities of categorical indicators), `"raincloud"` (each indicator's distribution by assigned profile), `"parallel"` (every case as a line) or `"pairs"` (scatter-plot matrix with each profile's 95% covariance ellipse).

For `plot_diagnostics()`: `"all"` (default), `"sizes"` (effective profile counts), `"posteriors"` (posterior probability of each case's assigned profile), `"entropy"` (each case's entropy relative to a flat posterior) or `"avepp"` (average posterior probability matrix). `"group_sizes"`, `"group_posteriors"`, `"group_entropy"` and `"group_avepp"` draw the same four views for the groups and their group classes; `"all"` includes them when the fit has two or more group classes, and with fewer they are refused with `latents_no_group_classes`. A fit with a noise component supports `"sizes"` and the group views only: `"all"` returns `"sizes"`, and the other individual-level views are refused with `latents_unsupported_noise`.

For `plot_selection()`: `"all"` (default), `"deltas"`, `"heatmap"`, `"smallest"`, `"trust"` or `"diagnostics"`, as described for `plot_enumeration()`.

For `plot_enumeration()`, three kinds of view; information criteria cannot be requested together with the other two:

- Information criteria: criterion columns of `as.data.frame(x)`, such as `"bic_groups"`, or `"all"`. On a profile or class grid the default draws AIC, both BICs and ICL over individuals; on a transition or family grid AIC, BIC and BIC over individuals; on a regression grid AIC, BIC, SABIC and ICL.
- Classification quality across the candidates, at each level with two or more classes: `"entropy"` (relative entropy), `"avepp"` (the lowest AvePP, with the cutoff dashed) or `"diagnostics"` (both).
- Model-selection evidence (none of which selects a model): `"deltas"` (the decrease in the criterion in `using` with each added profile and each added group class; see `get_results(x, "deltas")` and `"knees"`), `"heatmap"` (the criterion over profiles and group classes, relative to the lowest), `"smallest"` (the smallest class share, with 5% dashed) or `"trust"` (convergence, boundary and replication of every candidate). `"selection"` draws all four and `"diagnostics"`, as a list.

`category` — Which category's probability to draw: `"last"` (default), `"first"`, or a single label or index.

`cell_labels` — For sequence views, `TRUE` (default) prints each assigned profile in its cell when the cells are large enough for the text.

`n_group_classes`, `n_profiles`, `model`
: For `plot_tree()`, the trees and rows to draw. At the individual level a tree runs down the numbers of profiles (classes) at one number of group classes and one structure: `n_group_classes` and `model` choose trees, `n_profiles` the rows. At the group level a tree runs down the numbers of group classes at one number of profiles: `n_profiles` and `model` choose trees, `n_group_classes` the rows. `NULL` (default) keeps every converged candidate. A single count for the rows (`n_profiles` at the individual level, `n_group_classes` at the group level) is one level of every tree, and the plot compares that level's solutions instead: `plot_tree(x, n_profiles = 4, n_group_classes = c(2, 5))` draws the four-profile solution at two and at five group classes as two rows, each class in the colour of the class it matches in the first row.

`candidates`
: For `plot_tree()`, two or more candidate names drawn as the rows of one tree, in the order given, to compare those solutions. Not combined with `n_group_classes`, `n_profiles` or `model`.

`from`, `to`
: For `plot_tree()`, two solutions to compare as a two-row tree, each named by its class counts `c(n_profiles, n_group_classes)` (one count means one group class) or by its name: `plot_tree(x, from = c(3, 3), to = c(4, 3))`. Given together, and not with `candidates`. On a grid crossed with covariance structures, `model` names the structure of both.

`level`
: For `plot_tree()`, `"individuals"` (default; the cases divided into profiles or classes) or `"groups"` (the groups divided into group classes). For `plot()`, see *Arguments of* `plot()`.

`combine`
: `TRUE` (default) draws several criteria as panels of one plot; `FALSE` returns one plot per criterion. For `plot_tree()` (and `what = "tree"`), `FALSE` (default) returns one plot per tree and `TRUE` draws every tree as a panel of one plot.

`using`
: For `plot_enumeration(what = "deltas")`, `"heatmap"` or `"selection"`, the information criterion used. When not given, a profile or class grid uses `"bic_groups"` and the other grids `"bic"`; `"smallest"` and `"trust"` use none.

`cutoff`
: For `plot_enumeration(what = "avepp")`, `"diagnostics"` or `"selection"`, the AvePP cutoff drawn as a dashed line (default 0.8).

`mark_minimum`
: `TRUE` (default) circles each criterion's lowest value among converged candidates, without selecting a model. Information criteria only.

`...`
: With `what = "tree"` (on `plot()` of an enumeration or on `plot_enumeration()`), the arguments of `plot_tree()`. On `plot()` of a transition or group-class family grid, also `combine`, `labels`, `mark_minimum`, `cutoff` and `using`, passed to `plot_enumeration()`. Otherwise not used. Profile, class and transition models and enumerations refuse unknown arguments with `latents_bad_argument`; the other methods ignore them with a `latents_unused_argument` warning (see *Errors*).

Details

Returned plots can be printed, saved with `ggplot2::ggsave()`, or customized with `ggplot2::theme()`. Mixture-regression plots (only the `"fitted"` view for models that classify persons), pooled imputation plots, and bootstrap-test plots are displayed immediately, including inside loops, and return their ggplot objects invisibly.

Named functions provide access to related plot types through what: plot_profiles() displays profile summaries and indicator distributions; plot_responses() displays categorical response probabilities; plot_sequences() displays assigned profiles over occasions; plot_prevalence() displays profile proportions within group classes; plot_diagnostics() displays classification summaries; and plot_transitions() displays transition probabilities. For enumeration results, plot_enumeration() displays information criteria and classification summaries, plot_selection() displays model-comparison diagnostics, and plot_tree() displays correspondence between class solutions.

These functions use the corresponding plot(x, what =) method. Available views depend on the model family, as described below; what = "all" requests all supported views for the selected function. plot_views() lists plot types and their descriptions. For an enumeration object, plot_views(grid) also reports whether each view is supported and gives the reason when it is unavailable.

The enumeration functions support profile and class grids (enumerate_classes(), enumerate_lpa(), enumerate_lca()), transition grids (enumerate_classes(time =)), group-class family grids (enumerate_classes(famil =)) and regression grids (enumerate_regressions()). Available views are summarized below:

View	Profiles, classes	Transitions	Families	Regressions
criteria	14 columns	aic, bic, bic_individual	aic, bic, bic_individual	aic, bic, bic_
entropy, avepp, diagnostics	yes	yes	group classes	yes
deltas, heatmap, smallest, trust, selection	yes	yes	no	yes
tree	yes	yes	group classes	yes

Group-class family grids contain group classes without individual-level profiles. Their criteria and classification summaries are plotted against the number of group classes, and their trees compare group assignments (level = "groups", the default), with one tree per family. Views requiring both profiles and group classes are unavailable. Regression-grid trees use case-level posterior probabilities; the smallest-class view uses smallest_share, and the "trust" view omits boundary status because that column is unavailable. Unsupported views raise latents_unsupported_view.

Profile order (largest first), Okabe-Ito colours, and point shapes are consistent across views. Plotting requires the suggested package ggplot2.

Value

A ggplot object. what = "all" (on plot() of a fitted model or on a named plot function), plot_enumeration(combine = FALSE) with several criteria or classification-quality measures, and enumeration views when more than one is requested (including "selection") return a latents_plots list of ggplot objects, which draws each one when printed, pausing between plots on an interactive screen device. plot_tree() and what = "tree" return one ggplot for a single tree or combine = TRUE, and otherwise a latents_plots list named by tree. plot_views() returns a data.frame with one row per view: type (the value for what), group and description. plot_views(x) of a grid returns one row per what value of that grid: type, group ("criteria", "quality", "selection" or "tree"), supported (TRUE when the grid offers it), description and reason (why it is not offered; NA when it is).

Arguments of plot()

Besides x, what, main, subtitle and the arguments above, plot() takes these on the objects named:

`level` The confidence level for additive, mixture-regression and growth views (default 0.95). On pooled plots, a compatibility alias for `what`: `"profile"`, `"group"` or `"measurement"`.

`predictor` For the fitted view of a mixture regression, a numeric predictor in the formula. `NULL` (default) uses the first numeric predictor.

`time` For growth and trajectory views, the numeric time column. `NULL` (default) uses the first numeric variable in the fitted random or regression design.

`facet` For trajectories, `FALSE` (default) draws all classes in one panel; `TRUE` draws one panel per class.

`persons` For individual trajectories, the number of persons to show (default 12), or a vector of their identifiers.

`max_persons` The maximum number of observed trajectories drawn behind the class means (default 120); zero hides them.

`spread` For growth trajectories, the band of random-effect spread of persons around the class curve, from 0 (hidden) to less than 1; default 0.8.

`criterion` A compatibility alias for `what` in enumeration plots. On a profile grid it is used alone or with `what = "enumeration"`; transition and group-class family grids take `"bic"`, `"aic"` or `"bic_individual"` (or a unique abbreviation, such as `"bic_i"`), or any other view of the grid.

`palette` For bootstrap plots, a colour vector for the histogram and the observed statistic; `NULL` (default) uses the standard colours.

`style` An obsolete bootstrap plotting argument; ignored with a `latents_unused_argument` warning.

Views by object class

Each object takes `what =` for its views, `main`, `subtitle` and the arguments named here.

Fitted profile and class models From `multilpa()`, `multilca()`, `lpa()`, `lca()` and covariate models: `"profiles"` (the default; a fit whose indicators are all categorical with non-numeric category labels, which the profile summaries cannot score, draws `"responses"` by default), `"bars"`, `"heatmap"`, `"raincloud"`, `"parallel"`, `"pairs"` (each profile's 95% covariance ellipse), `"responses"`, `"probabilities"` (not for covariate models), `"sequences"` (fits with `time`; `cell_labels = TRUE` prints the profile in each cell), `"sizes"`, `"entropy"`, `"posteriors"`, `"avepp"` and `"all"`. The profile views name in a caption any ordinal, count or (for the means) categorical indicator they omit.

Transition fits From `lta()`: `"transitions"` (default), then `"profiles"`, `"bars"`, `"heatmap"`, `"responses"`, `"sequences"`, `"sizes"`, `"entropy"`, `"posteriors"`, `"avepp"` and `"all"`, without `statistic`; the profile, bar and response views draw the 95% intervals of the transition fit's `parameter_inference()` (`intervals`). A general `lta()` fit takes no `what` and draws its transition matrix, which `plot_transitions()` draws too.

Enumerations Criterion columns or `"all"` (with `combine`, `labels` and `mark_minimum`), `"tree"` (with the arguments of `plot_tree()`), the classification-quality views `"entropy"`, `"avepp"` and `"diagnostics"` (with `cutoff`), and the model-selection views `"deltas"` and `"heatmap"` (which use the criterion in `using`), `"smallest"` and `"trust"`; `"selection"` draws those four and `"diagnostics"`. Transition and group-class family grids draw `"bic"` by default and take every view they offer (the table under *Description*); a regression grid draws its default criteria and takes the same arguments as `plot_enumeration()`.

Additive fits `"means"` (default; each group class's mean with its Wald interval), `"variances"` (within- and between-group variances) or `"intercepts"` (each group's posterior intercept); `level` sets the interval. `plot_profiles()` draws `"means"`.

Cross-level fits `"profiles"` (default), `"group_means"` or `"composition"` (profile shares within each group class). `plot_profiles()` draws `"profiles"` and `plot_prevalence()` `"composition"`.

Mixture regressions `"fitted"` (default: the outcome against a numeric `predictor`, one line per class), `"coefficients"` or `"posteriors"`; `level` sets the intervals. Fits that classify persons (`class_level = "group"`) default to `"trajectories"` (unless `predictor` is given) and also take `"individuals"` and `"classification"`, as growth mixtures do. `plot_profiles()` draws `"fitted"`, or `"trajectories"` for fits that classify persons, and `plot_diagnostics(what = "posteriors")` draws `"posteriors"`.

Growth mixtures `"trajectories"` (default: class mean trajectories with bands; `facet`, `max_persons`, `spread`), `"individuals"` (`persons`: a number or identifiers), `"coefficients"`, `"random"` (predicted random effects with each class's 95% ellipse) or `"classification"` (also `"posteriors"`); `time` names the time variable, `level` the band. `plot_profiles()` draws `"trajectories"` and `plot_diagnostics(what = "posteriors")` draws `"posteriors"`.

Pooled imputations `"profile"` (default; the probability of each profile within each group class), `"group"` or `"measurement"` (profile means and variances) parameters: each imputation's estimate beside the pooled one with its interval. `plot_profiles()` draws `"measurement"` and `plot_prevalence()` draws `"profile"`.

Model comparisons, bootstrap tests, diagnostics One view each: the criteria of each compared model, the simulated null distribution with the observed statistic, and the views of `plot_diagnostics()`.

Named plot functions on other fitted models

Named plotting functions dispatch to the corresponding view for each model family. For example, `plot_profiles(fit)` and `plot(fit, what = "trajectories")` produce the same growth-mixture plot:

Growth mixtures, and mixture regressions that classify persons (`class_level = "group"`) `plot_profiles()` draws `"trajectories"` (each class's curve over time) and `plot_diagnostics(what = "posteriors")` draws `"posteriors"` (each person's class probabilities).

Other mixture regressions `plot_profiles()` draws `"fitted"` (each class's line against the first numeric predictor) and `plot_diagnostics(what = "posteriors")` draws `"posteriors"`.

Additive and dispersion fits `plot_profiles()` draws `"means"` (each group class's means across indicators).

Cross-level fits `plot_profiles()` draws `"profiles"` and `plot_prevalence()` draws `"composition"` (profile shares within each group class).

Pooled imputations `plot_profiles()` draws `"measurement"` (the pooled profile means and variances beside each imputation's) and `plot_prevalence()` draws `"profile"` (the pooled probability of each profile within each group class).

General `lta()` fits `plot_transitions()` draws the transition matrix.

Every other combination, including the `what` values of `plot_profiles()` other than `"lines"` and those of `plot_diagnostics()` other than `"posteriors"`, is refused with `latents_nothing_to_plot` naming the views that fit's `plot()` offers; `what = "all"` draws the mapped views and names the others in a message. These functions pass on only the arguments given, and refuse with `latents_bad_argument`

one the mapped view does not take (for example `intervals` or `scale` for a growth trajectory); use `plot()` for that view's own arguments, such as `time`, `facet` or `predictor`, and for the views no named function draws.

Compatibility arguments

Enumeration plots accept `criterion` as an alternative to `what` (on a profile grid, `what = "enumeration"` requests the default criteria view). Pooled plots accept `level` as an alternative to `what`, and bootstrap plots accept `palette` (the histogram and observed-line colours). On an enumeration, `criterion` with any `what` other than `"enumeration"` is an error of class `latents_bad_argument` (with `"tree"`, a warning, as the tree uses no criterion); `level` together with `what` on a pooled plot is an error of the same class.

Errors

`latents_missing_package` without ggplot2; `latents_bad_argument` for an argument the object's plot does not take, except that the plots of additive, cross-level, general transition (`lta()`), growth-mixture, mixture-regression, pooled, comparison and bootstrap results and of transition or group-class family grids ignore it with a `latents_unused_argument` warning (including a bootstrap plot's `style`); `latents_bad_argument` also when a named plot function is given an object it does not draw (a named fit function given anything other than a fitted model, or a grid function given anything other than an enumeration); `latents_nothing_to_plot` (also of class `latents_bad_argument`) when a named fit function, or its `what`, has no view of that fitted model, naming the views its `plot()` offers (for example `plot_transitions()` on a model without transitions, or `plot_diagnostics(what = "sizes")` on a growth mixture); `latents_nothing_to_plot` when the fit lacks what a view needs (for example posteriors or entropy of a one-profile fit, prevalence of a covariate model, or the case-level views of a centred fit); `latents_unknown_category` when `category` names no category of an indicator; `latents_no_continuous`, `latents_no_categorical` and `latents_no_time` when a view needs indicators or occasions the fit does not have; `latents_unknown_criterion` (also `latents_bad_argument`) for a criterion not in the grid; `latents_unsupported_view` (also `latents_bad_argument`) for a view the kind of grid does not offer; `latents_no_tree` (also of class `latents_nothing_to_plot`) when a tree would have fewer than two rows, or at the group level of a grid with one group class; `latents_unknown_candidate` when `candidates`, `n_group_classes`, `n_profiles` or `model` names something no converged candidate has; `latents_failed_candidate` when a named candidate was not fitted; `latents_unconverged` when a named candidate did not converge; `latents_unsupported_noise` for a tree of a fit with a noise component.

References

Satopaa, V., Albrecht, J., Irwin, D. and Raghavan, B. (2011). Finding a "kneedle" in a haystack: detecting knee points in system behavior. *31st International Conference on Distributed Computing Systems Workshops*, 166-171.

Zappia, L. and Oshlack, A. (2018). Clustering trees: a visualization for evaluating clusterings at multiple resolutions. *GigaScience*, 7(7), giy083.

Examples

```
plot_views()
if (requireNamespace("ggplot2", quietly = TRUE)) {
```

```
  fit <- multilpa(course_engagement, c("browse", "lectures", "forum_read"),
                  "student", n_profiles = 2, n_group_classes = 2,
                  n_starts = 2, seed = 1)
  plot_profiles(fit)
  plot_diagnostics(fit, what = "avepp")
}
```

`lca` *Fit a latent class model*

Description

Latent class analysis describes heterogeneity in categorical response patterns through a specified number of latent classes. Each class is characterized by the probabilities of the response categories for each indicator. These probabilities describe how response patterns differ between classes.

`lca()` estimates the class-specific response probabilities and class proportions, together with posterior membership probabilities for each observation. Indicators are assumed independent conditional on class membership unless associations between selected indicators are explicitly specified. The posterior probabilities describe uncertainty in classification; an observation's assigned class is a summary of these probabilities.

The model treats observations as independent. For observations nested within groups, use `multilca()`. Ordinal and count indicators can also be fitted using the structured measurement models described in `multilpa()`.

Usage

```
lca(data, vars, n_classes, ...)
```

Arguments

`data`	A data frame with one row per observation.
`vars`	Names of the discrete indicator columns. Categories are taken from the data. Columns named in `ordinal` or `count` (through `...`) are modelled as such; the rest are categorical.
`n_classes`	Number of classes; a positive integer.
`...`	Further arguments for `multilca()`, such as `missing`, `n_starts`, `seed`, `min_probability` and `profile_covariates`, except `id` and `n_group_classes`.

Value

A fitted object as returned by `multilca()`, usually of class `multilpa`, or `multilpa_covariates` with membership covariates. Its `call` records the original `lca()` call. Classes are labelled `profile_1`, `profile_2`, and so on in the returned tables.

Conditions

`latents_bad_argument` when `id`, `n_group_classes` or `n_profiles` is passed; the conditions `multilca()` raises otherwise.

References

Lazarsfeld, P. F., & Henry, N. W. (1968). *Latent Structure Analysis*. Houghton Mifflin.

See Also

`lpa()` for continuous indicators; `multilca()` for the two-level model; `enumerate_lca()` to compare numbers of classes.

Examples

```
fit <- lca(student_esm, c("time_with_friends", "on_social_media",
                          "tv_video_games", "sports"),
           n_classes = 2, n_starts = 3, seed = 1)
fit
plot(fit, what = "heatmap")
```

---

lpa *Fit a latent profile model*

Description

Latent profile analysis describes heterogeneity in continuous indicators through a specified number of latent profiles. Each profile represents a pattern of indicator means and within-profile variation. `lpa()` estimates these patterns using a finite mixture of multivariate normal distributions.

By default, indicators are independent within profiles and their variances differ across profiles. Alternative covariance structures allow associations between indicators within profiles and equality constraints across profiles. The fitted model provides profile means, variances, proportions, and posterior membership probabilities for each observation, together with model-fit and classification summaries.

The model treats observations as independent. For observations nested within groups, use `multilpa()`. Additional measurement options support mixed continuous and discrete indicators.

Usage

```
lpa(data, vars, n_profiles, ...)
```

Arguments

`data`	A data frame with one row per observation.
`vars`	Names of the indicator columns. Numeric columns are continuous indicators; categorical ones are named in `categorical`.
`n_profiles`	Number of profiles; a positive integer.
`...`	Further arguments for `multilpa()`: the covariance structure (`model`, or `variance_model` and `covariance_model`, or `volume`, `shape` and `orientation`), `missing`, `categorical`, `n_starts`, `seed`, `boundary`, `prior`, `noise` and `profile_covariates`. Arguments `id` and `n_group_classes` are not accepted.

Value

A fitted object as returned by `multilpa()`, usually of class `multilpa`, or `multilpa_covariates` with membership covariates. Its `call` records the original `lpa()` call.

Conditions

`latents_bad_argument` when `id` or `n_group_classes` is passed; the conditions `multilpa()` raises otherwise.

References

Oberski, D. (2016). Mixture models: Latent profile and latent class analysis. In J. Robertson & M. Kaptein (Eds.), *Modern Statistical Methods for HCI* (pp. 275–287). Springer.

See Also

`lca()` for categorical indicators; `multilpa()` for the two-level model; `enumerate_lpa()` to compare numbers of profiles.

Examples

```
fit <- lpa(srl, c("cognitive_strategies", "intrinsic_value",
                  "self_efficacy", "self_regulation", "test_anxiety"),
           n_profiles = 2, variance_model = "equal",
           covariance_model = "full", n_starts = 3, seed = 1)
fit
plot(fit, what = "raincloud")
```

`lta` *Fit a latent transition model*

Description

Latent transition analysis examines stability and change in latent states across repeated observations. States are defined by patterns in continuous or discrete indicators, and transition probabilities describe the probability of remaining in a state or moving to another between successive occasions. `lta()` estimates the measurement and transition parameters jointly, accounting for uncertainty in state membership.

By default, transitions follow a first-order, time-homogeneous model: the state at an occasion depends on the immediately preceding state, and transition probabilities are constant across occasions. Measurement parameters are also equal across occasions, providing a common definition of each state over time.

Additional latent group classes can represent differences between individuals or other units in their initial-state distributions and transition patterns. Extensions allow covariate-dependent transitions, occasion-specific parameters, and second-order dependence. The fitted model provides measurement estimates, initial-state probabilities, transition probabilities, and posterior membership probabilities.

Usage

```
lta(
  data,
  vars,
  id,
  n_profiles,
  time,
  n_group_classes = 1L,
  variance_model = c("varying", "equal"),
  n_starts = 10L,
  max_iter = 5000L,
  tol = 1e-10,
  min_variance = NULL,
  seed = NULL,
  missing = c("error", "fiml"),
  covariance_model = c("diagonal", "full"),
  categorical = character(),
  min_probability = 1e-10,
  occasions = c("observed", "grid"),
  select_start = c("likelihood", "converged"),
  transitions = c("homogeneous", "occasion"),
  transition_covariates = character(),
  initial_covariates = character(),
  measurement = c("invariant", "occasion"),
  order = 1L,
  model = NULL,
  mover_stayer = FALSE,
  weights = NULL,
  ordinal = character(),
  count = character(),
  count_model = c("poisson", "negative_binomial"),
  count_dispersion = c("varying", "equal"),
  drop_empty = TRUE,
  search = c("kmeans", "emEM"),
  n_screen = NULL,
  acceleration = c("squarem", "none"),
  cores = 1L
)
```

Arguments

`data`	A data frame containing the indicators and the group identifier.
`vars`	Unique character vector of indicator column names. Columns not named in `categorical`, `ordinal` or `count` are continuous indicators.
`id`	Name of the observed group identifier column. Character, factor and numeric identifiers are supported; missing identifiers are not. Required; `id = NULL` is not accepted.

`n_profiles` Integer number of individual profiles, at least two (`latents_bad_transition` otherwise).

`time` Name of the column giving each observation's occasion within its group. Required. Values must be complete and unique within each group. They are ordered as `sort()` orders them: numbers numerically, a factor by its levels, and character strings alphabetically. Numbers stored as text whose alphabetical order is not their numeric order (`"2"`, `"10"`), and numbered labels sharing their text around the number whose alphabetical order is not the numbers' (`"T2"`, `"T10"`; `"week 2"`, `"week 10"`), raise `latents_bad_time` (also `latents_bad_data`); zero-padded numbers and ISO dates are accepted.

`n_group_classes` Positive integer number of latent group classes (default 1). One gives an ordinary latent transition model; more than one fits a mixture of transition models over groups.

`variance_model` `"varying"` (the default; profile-specific variances or covariance matrices) or `"equal"` (shared across profiles).

`n_starts` Positive integer number of EM starts (default 10). The first start is deterministic: Ward's hierarchical clustering of the standardized indicators (on at most 2000 evenly spaced rows), each cluster initialized with its own means and variances, and, with more than one group class, Ward's clustering of the groups' profile compositions. The remaining starts are k-means from random centres. A supplied `start` replaces the first start, and the remaining starts are random. With `start` and `max_iter = 0`, exactly one start is evaluated regardless of `n_starts`. Zero-weight units do not contribute to initialization. For `search = "emEM"`, `n_starts` is the number of final optimization runs, including the protected first start (see `search`).

`max_iter` Nonnegative integer maximum number of EM updates per start (default 5000, higher than the 1000 of `multilpa()` to match the tighter `tol`). A fit that stops short warns with `latents_unconverged`. `max_iter = 0` updates nothing and evaluates the starting values.

`tol` Positive convergence tolerance (default `1e-10`). EM convergence is assessed from the change in log likelihood over one step or SQUAREM cycle, with threshold `tol` times the number of observed indicator values in positive-weight rows. This default is stricter than the `1e-8` default of `multilpa()`.

`min_variance` Lower bound on each variance, or on each covariance eigenvalue under full covariance; the fit is then constrained maximum likelihood. `NULL` (the default) sets it to `1e-6` times the smallest sample variance of the continuous indicators (observed values in rows of positive weight, on the scale fitted, after any centring and scaling), so that the default bound adapts to the measurement scale. A numeric value specifies an absolute bound in squared indicator units. The bound used is stored as `min_variance` on the fit. Estimates at the bound are reported and raise a warning.

`seed` Optional random seed: any integer `set.seed()` accepts, including negative integers. When supplied, the caller's random-number state is restored on exit.

`missing` `"error"` (the default) rejects missing indicators; `"fiml"` maximizes the observed-data likelihood under an ignorable missingness mechanism (MAR), integrating

missing indicators out rather than filling them in. This also holds with membership covariates; the covariates themselves must be complete (`latents_bad_data`).

`covariance_model`
: `"diagonal"` (the default) assumes conditional independence; `"full"` estimates within-profile residual covariances.

`categorical`
: Character vector naming indicators to treat as categorical. Each is modelled by unrestricted, profile-specific response probabilities over its observed categories (the latent class measurement model); binary, ordinal and unordered indicators are handled by the same parameterization. Numeric, integer, logical, character and factor columns are accepted. Indicators not named here stay Gaussian, so naming a subset fits a mixed-mode model.

`min_probability`
: Positive lower bound on every categorical response probability (default `1e-10`), constraining the maximum-likelihood problem as `min_variance` does for Gaussian indicators.

`occasions`
: How a group's occasions are placed on the transition grid. `"observed"` (the default) numbers each group's own observations consecutively, so a transition always links two adjacent observations. `"grid"` places them on the grid of every position seen in the data, so a group that skips a position still makes a transition across the gap and contributes no measurement information at it. The two agree when every group is observed at every position. Under either, a group's first observation is its first occasion: occasion-specific transitions and measurement count occasions from each group's own start, not from the first position in the data.

`select_start`
: Rule for selecting a fitted solution. `"likelihood"` (the default) selects among starts within `tol * max(1, n_observed)` of the highest log likelihood, preferring converged starts within this set. Here `n_observed` counts observed indicator values in positive-weight rows. `"converged"` restricts selection to converged starts when any are available; otherwise it uses all starts. Remaining ties are resolved by start order. Convergence is required for standard errors but does not establish identification or global optimality. `summary()` and `get_results(fit, "starts")` report all starts.

`transitions`
: `"homogeneous"` (the default) uses one transition matrix for every occasion; `"occasion"` estimates a separate one for each move (occasion 1 to 2, 2 to 3, ...). A move into an occasion at which no row observes an indicator (`missing = "fiml"`) is not identified: after the last observed occasion it does not enter the likelihood, and before that occasion it enters only through its product with the move out. Its logits are then reported with `estimated = FALSE` and no standard error, and are not counted in `n_parameters`, AIC or BIC; likewise the second-order logits when nothing is observed from occasion 3 on, and the measurement block of an empty occasion under `measurement = "occasion"`. Under either setting, a profile that is never occupied before a later occasion (in a group class) has no move out of it: its outgoing logits are fixed at zero (equal probabilities), flagged `estimated = FALSE`, not counted, and `latents_empty_transition_row` is raised, as the homogeneous model reports its uniform row.

`transition_covariates`
: Names of columns whose values shift the transition probabilities through a

multinomial logit per origin profile: the log odds of moving to each other profile rather than staying. A column may change over occasions; the value at the destination occasion is used. Factors are expanded to indicator columns, after dropping levels the data do not use (as `stats::glm()` does). With `occasions = "grid"`, a move into a position the group skipped has no row to read the covariates from, and the fit raises `latents_bad_data`; use `occasions = "observed"`, or add the skipped rows with their covariate values (their indicators may be missing under `missing = "fiml"`).

`initial_covariates`
: Names of columns that shift the initial profile distribution (a multinomial logit, reference: the last profile), read at each group's first occasion. Factors are expanded as for `transition_covariates`.

`measurement`
: `"invariant"` (the default) keeps the profiles' means, variances and response probabilities equal across occasions; `"occasion"` estimates them separately at each occasion.

`order`
: `1` (the default) or `2`. With `2`, the profile at each occasion from the third on depends on the profiles at the two previous occasions (a second-order Markov chain); the first transition remains first order.

`model`
: A covariance structure as `multilpa()` takes it (`"EEI"`, `"VVI"`, `"VEI"`, ... ; mclust's codes). Standard errors are given for every structure with the extensions (covariate, occasion-varying or second-order transitions, occasion-specific measurement), and for EEI, VVI, EEE and VVV in the homogeneous model, which rejects the other ten with `latents_unsupported_inference`.

`mover_stayer`
: `FALSE` (the default), or `TRUE` to add a class of stayers: groups that never change profile (identity transitions) and have their own initial profile distribution (Goodman's mover-stayer model). `n_group_classes` then counts the mover classes; the stayer class is reported as `"stayers"`.

`weights`
: `NULL`, or the name of a column of `data` holding one sampling weight per `id` (constant over its occasions). As in `multilpa()`: pseudo maximum likelihood with the weights scaled to sum to the number of `id` units, sandwich standard errors by default, and integer weights equal to repeating each sequence. A weighted fit uses the general transition engine and returns a `multilpa_lta`, whose model-implied `"transitions"` and `"initial"` tables average the groups with their weights.

`ordinal`, `count`, `count_model`, `count_dispersion`
: Ordinal and count indicators, as in `multilpa()`; with `measurement = "occasion"` their parameters are estimated per occasion. These estimates are returned by `get_results(fit, "ordinal")` and `get_results(fit, "count_means")`. A fit with them uses the general transition engine.

`drop_empty`
: `TRUE` (the default) or `FALSE`: how the reported statistics treat rows whose every indicator is missing, and groups consisting only of such rows. This setting affects reporting, not estimation. With `TRUE` they are excluded, as in Mplus, from the individual and group sample sizes of the information criteria, from relative entropy, AvePP and the entropy term of ICL, AWE and CLC, from the classification tables and diagnostic plots, and from the effective class counts. Row-aligned assignment and posterior tables retain these units but report their classifications and posterior probabilities as `NA`. A partially observed row remains

classified, and an empty row in an informative group can retain its group-class assignment. With FALSE every row and group is counted in those statistics, except in the individual sample size of a multilpa() fit, which excludes such rows either way. Units of sampling weight zero are excluded from every count and entropy term under either setting. compare_models() and bootstrap_lrt() do not compare fits made with different drop_empty. The setting is stored on the fit and passed on by enumerations and refits; it has no effect on data without such rows.

search — How the starts are found: "kmeans" (the default) or "emEM", as described for multilpa(). Unlike multilpa(), lta() does not choose its default by the covariance structure.

n_screen — For search = "emEM", the number of starts screened, at least n_starts. NULL (the default) screens five times n_starts. The protected first start is included in the n_starts final optimization runs. Increasing n_screen does not increase that number. A single supplied start with n_starts = 1 disables screening. A value with search = "kmeans", or one below n_starts, raises latents_bad_argument.

acceleration — "squarem" (the default) accelerates EM with SQUAREM (Varadhan & Roland, 2008), as multilpa() does: each cycle extrapolates along the last two EM steps, including the initial and transition probabilities, and ends with an ordinary EM step. An extrapolated point that would lower the likelihood, or that leaves a probability non-positive or a row not summing to one, is replaced by the plain EM point, preserving monotone likelihood updates. Convergence is assessed over a complete cycle. "none" runs ordinary EM. iterations and max_iter count EM steps under either setting.

cores — Number of processor cores for the random starts (default 1). The starting values are drawn in order in the current process and only the EM runs are distributed, so with seed given the result is identical to cores = 1. Under search = "emEM" both screening and the final runs are distributed. Workers are forked processes on macOS and Linux, and a socket cluster on Windows and on macOS with Apple's Accelerate BLAS, as in enumerate_classes(). Lost worker results are retried once with half as many workers (rounded up), with one latents_worker_lost warning reporting losses and recovery and suggesting fewer cores. A start still lost is recorded as failed; the fit uses the remaining starts. If all starts fail, the usual latents_all_starts_failed error is raised. Socket workers use bounded chunks; completed tasks within a failed chunk may also be retried. To fit many models, give cores to enumerate_classes(), which runs starts across candidates in one pool.

Details

Under the default first-order model, the next profile depends on the current profile and, when present, the latent group class. Earlier profiles provide no additional dependence conditional on these variables. Homogeneous transitions use the same probabilities at each occasion. Measurement parameters are shared across occasions and across group classes. A profile that no group occupies before its final occasion leaves its transition row without information; the row is then uniform by construction, not estimated, and its probabilities are not counted in n_parameters, AIC or BIC. Such a row raises a warning when the model is fitted, is flagged by the estimated column

of the transition table and is listed by `get_results(fit, "transitions", estimated = FALSE)`. Profile labels are arbitrary and are not comparable across fits without alignment.

With `max_iter = 0`, the model evaluates the starting values without updating them and reports `converged = FALSE`. Posterior probabilities, expected transition counts and implied profile prevalences are computed at those starting values.

Value

A `multilpa_transitions` object containing `means`, `variances`, optional `covariances` and `response_probabilities`, `initial_probabilities` (group classes by profiles), `transition_probabilities` (profiles by profiles by group classes, rows indexing the profile moved from), `group_probabilities`, posterior matrices, classifications, log likelihood, information criteria, restart diagnostics and convergence history. `get_results()` returns the tidy transition table and the other tables, `parameter_inference()` Wald or bootstrap standard errors, and `bootstrap_lrt()` compares nested fits. Global optimality is not guaranteed.

With `transitions = "occasion"`, `transition_covariates`, `initial_covariates`, `measurement = "occasion"`, `order = 2`, `mover_stayer = TRUE`, `weights`, `ordinal` or `count`, the result is a `multilpa_lta` object, read with `get_results()`: transition and initial logit coefficients with Wald standard errors (observed, robust or OPG, from analytic scores), model-implied transition probabilities per occasion (with an `estimated` column, `FALSE` for a row resting on a logit the likelihood does not identify; a move only units of weight zero make is left out), profiles per occasion, and the assignments in the layout every fit shares (taking `data` and `truth`). Any other argument to its tables raises `latents_bad_argument`, and `summary()` collects them (taking `level` and `vcov_type`). Missing indicators (`missing = "fiml"`) and covariance structures are supported; diagonal fits are finished by a quasi-Newton search on the exact likelihood, others by EM.

References

Collins, L. M., & Lanza, S. T. (2010). Latent class and latent transition analysis. Wiley.

Varadhan, R., & Roland, C. (2008). Simple and globally convergent methods for accelerating the convergence of any EM algorithm. Scandinavian Journal of Statistics, 35, 335–353.

Vermunt, J. K. (2003). Multilevel latent class models. Sociological Methodology, 33, 213–239. doi:10.1111/j.0081-1750.2003.t01-1-00131.x.

See Also

`get_results()` for the fitted transition probabilities and for the assignments in order, and `multilpa()` for the cross-sectional model this one shares its measurement parameters with.

Examples

```
# Model engagement transitions across each student's course sequence.
fit <- lta(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student", n_profiles = 2, time = "sequence", n_starts = 2, seed = 1
)
get_results(fit, what = "transitions")
summary(fit)
```

`mixture_regression` *Estimate regression relationships that vary across latent classes*

Description

`mixture_regression()` examines whether the relationship between an outcome and its predictors differs across latent classes. It represents the conditional outcome distribution as a mixture of class-specific regression models, allowing classes to differ in their outcome levels, predictor associations, or trajectories over time.

Regression parameters and class-membership probabilities are estimated jointly by maximum likelihood. Coefficients may vary across classes or, for selected predictor terms, be constrained to a common value. Predictors in the outcome formula describe variation within classes; covariates supplied through `membership` describe variation in the probability of belonging to each class. Posterior membership probabilities express the uncertainty in assigning observations or groups to the estimated classes.

The function supports Gaussian, binomial, Poisson, negative-binomial, and ordinal outcome models, subject to the restrictions described below. Class membership may be defined for individual observations or shared across repeated observations from the same unit. With time predictors, shared membership defines class-specific trajectories; Gaussian random effects additionally allow individual trajectories to vary within classes.

Usage

```
mixture_regression(
  formula,
  data,
  n_classes,
  family = c("gaussian", "binomial", "poisson", "negative_binomial", "ordinal"),
  id = NULL,
  class_level = c("observation", "group"),
  n_group_classes = 1L,
  common = NULL,
  membership = ~1,
  group_membership = ~1,
  variance = c("varying", "equal"),
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  min_variance = 1e-06,
  seed = NULL,
  missing = c("error", "omit"),
  select_start = c("likelihood", "converged"),
  vcov_type = c("observed", "robust", "opg", "none"),
  weights = NULL,
  random = NULL,
  random_covariance = c("varying", "equal", "proportional"),
```

```
  random_diagonal = FALSE,
  cluster = NULL,
  residual_correlation = c("none", "ar", "brownian"),
  residual_time = NULL
)
```

Arguments

formula	A model formula `outcome ~ predictors`. Factors, interactions and `offset()` are supported.
data	A data frame.
n_classes	Number of regression classes.
family	`"gaussian"` (default), `"binomial"`, `"poisson"`, `"negative_binomial"` or `"ordinal"` (proportional-odds cumulative logit; see *Families*).
id	`NULL` (default), or the name of the column identifying groups (persons, schools, ...).
class_level	`"observation"` (default; each row has its own class) or `"group"` (all rows of a group share a class; requires `id`).
n_group_classes	Number of second-level group classes (default 1, no latent group classes), for `class_level = "observation"` with `id`, or for the clusters of a multilevel growth mixture (`cluster`).
common	`NULL`, or the predictor terms whose coefficients are shared by every class: variable names such as `"age"`, or a one-sided formula such as `~ age`. Each term must appear in `formula`. The intercept cannot be shared: `"intercept"` or `~ 1` alone raises `latents_bad_argument`.
membership	Covariates predicting class membership (a multinomial logit, the first class as the reference): variable names such as `"age"`, or a one-sided formula (`~ 1`, the default, for none). Row-level for `"observation"`; constant within groups for `"group"`.
group_membership	Group-level covariates predicting the group class, for the two-level model, as names or a one-sided formula. Must be constant within groups.
variance	`"varying"` (default; a residual standard deviation, or a negative-binomial dispersion, per class) or `"equal"` (one shared). Gaussian and negative-binomial families only.
n_starts	Number of random starts (default 10), besides one start built from the residuals of a pooled regression. Each start runs for up to 50 EM iterations; the better half (at least two) continue to convergence.
max_iter	Maximum EM iterations per start (default 1000).
tol	Convergence tolerance (default `1e-8`): EM has converged when one step changes the log likelihood by no more than `tol` times the number of positive-weight rows. The tolerance is therefore expressed per observation.

`min_variance` Floor for a Gaussian residual variance, relative to the outcome's variance (default `1e-6`). Starts in which a class reaches the lower bound are excluded from selection. In a growth mixture, the same floor bounds each random effect's conditional variance, applied in the standardized coordinates the model is fitted in (outcome and predictors centred and scaled), so it does not depend on the units of the time variable.

`seed` `NULL` (default) advances the caller's random stream. An integer seed restores the caller's `.Random.seed` afterwards. R does not store a cached Box-Muller normal draw in `.Random.seed`, so that hidden cached draw cannot be preserved.

`missing` `"error"` (default) rejects rows with missing values in any variable the model uses; `"omit"` drops them with a `latents_rows_dropped` warning stating how many.

`select_start` `"likelihood"` (default) selects the start with the highest likelihood; `"converged"` prefers the best start that converged.

`vcov_type` Covariance of the estimates stored with the fit: `"observed"` (default; inverse observed information), `"robust"` (sandwich with the CR1 factor `G / (G - 1)`, clustered on the top-level unit, or on `id` for a single-level fit given `id`) or `"opg"` (outer product of the scores). `"none"` skips inference.

`weights` `NULL`, or the name of a numeric column of `data` with a sampling weight per independent unit: per row without `id`, per `id` group otherwise (constant within it). Pseudo maximum likelihood with the weights scaled to sum to the units of positive weight; `vcov_type` defaults to `"robust"` and rejects `"observed"` and `"opg"`. The information criteria use the weighted log likelihood, and the entropy term of ICL is weighted by the sampling weights, as the likelihood is. A unit of weight zero is absent from the fit: from every start, estimate, sample size and criterion; it keeps its row and posterior. A row of weight zero whose factor level (or ordinal outcome category) no row of positive weight has is left out, with a `latents_rows_dropped` warning, since the model has no parameter for that level.

`random` `NULL` (default; no random effects), or the random effects within classes: variable names such as `"time"` (a random intercept and a random slope on `time`) or `"intercept"` (random intercepts only), or a one-sided formula such as `~ 1 + time`. A slope without a random intercept needs the formula `~ 0 + time`. See *Growth mixture models*.

`random_covariance`
How the random-effect covariance differs across classes: `"varying"` (default; one per class), `"equal"` (one shared matrix, as `lcmm::hlme()` with `nwg = FALSE`) or `"proportional"` (a shared matrix times a class-specific scale, the last class's being one; `nwg = TRUE`).

`random_diagonal`
`TRUE` for uncorrelated random effects (a diagonal covariance, as `idiag = TRUE` in lcmm); default `FALSE`.

`cluster` `NULL` (default), or the name of a column grouping the persons (`id`) into clusters, such as schools, for the multilevel growth mixture model (with `random`): each cluster belongs to one of `n_group_classes` group classes, which sets the trajectory-class probabilities of its persons. The clusters are the independent

units of the likelihood, the standard errors and the BIC; with `n_group_classes = 1` the model is the single-level growth mixture with clusters as those units, comparable by BIC with fits with additional group classes. See *Growth mixture models*.

`residual_correlation`
: `"none"` (default) for independent residuals within a person; with `random`, `"ar"` adds a continuous-time autoregressive process, covariance `sd^2 exp(-rate |t - s|)`, and `"brownian"` Brownian motion, `sd^2 min(t, s)`, to the residual variance, shared by the classes: `lcmm::hlme(cor = AR(time))` and `cor = BM(time)`. The model without the process is fitted first from every start; the process is then estimated jointly with all other parameters. A process the data cannot separate from the residual variance or the random effects is left on its bound and warned about (`latents_random_boundary`); its rows then have no standard error. Refused with `latents_bad_argument` without `random`.

`residual_time`
: Name of the numeric time column for the residual process. Defaults to the first numeric variable in `random`, or otherwise the predictors. Brownian motion requires non-negative times (`latents_bad_data`).

Value

An object of class `latents_mixture_regression`, or `latents_growth_mixture` when `random` is given. `as.data.frame()` returns the coefficient table and `get_results()` every table by name (described there); `predict()`, `plot()`, `summary()`, `coef()`, `vcov()`, `confint()`, `logLik()` and `nobs()` also apply.

Interpretation

The classes describe heterogeneity under the specified regression model. Their interpretation depends on the outcome distribution, included predictors, coefficient constraints, and treatment of repeated observations. Within-class coefficients describe conditional associations and do not, by themselves, establish causal effects. Estimation stability and classification uncertainty should be examined before assigning substantive meaning to the classes.

Nesting

Arguments `class_level` and `n_group_classes` specify the level of latent class membership and the likelihood contribution of each independent unit.

Single level (`id = NULL`) Each row is an independent unit with its own class.

`class_level = "group"` All rows of a group (`id`) belong to the same class: the regression mixture for repeated measures, as `flexmix(y ~ x | id)`. A group's likelihood contribution is the product of its rows' densities.

`class_level = "observation"` **with** `id` **and** `n_group_classes >= 2` Each row has its own class, and a second-level group class determines the class probabilities within its groups (Vermunt 2003). With `n_group_classes = 1`, `id` enters only the `"robust"` standard errors, which are then clustered on `id`.

Growth mixture models

For repeated measurements (`id`, `class_level = "group"`), including time in `formula` specifies class-specific trajectories. Without random effects, this is a latent class growth model (Nagin, 2005). Argument `random` adds within-class variation in individual trajectories, yielding a growth mixture model (Verbeke and Lesaffre, 1996; Muthen and Shedden, 1999). For individual i in class k, the model is

$$y_i = X_i\beta_k + Z_i b_i + e_i, \quad b_i \sim N(0, G_k), \quad e_i \sim N(0, \sigma_k^2 I),$$

where `Z` is the design matrix specified by `random`. The Gaussian random effects are integrated analytically. Estimation uses EM followed by quasi-Newton optimization with analytic scores. The fitted object has class `latents_growth_mixture`; its tables and plots are described on `get_results()` and plot(). Random effects require the gaussian family, `id` and `class_level = "group"`.

With `residual_correlation`, the residuals also carry a process shared by the classes, $e_i \sim N(0, W_i + \sigma_k^2 I)$: a continuous-time autoregressive process (`"ar"`) or Brownian motion (`"brownian"`) over `residual_time`, corresponding to the correlation structures specified by `lcmm::hlme(cor = )`.

With `cluster`, persons are nested in clusters (students in schools) and the multilevel growth mixture model is fitted: each cluster belongs to a group class h, and the trajectory-class logits of its persons have a group-class-specific intercept and shared covariate slopes,

$$L = \prod_j \sum_h \omega_h \prod_{i \in j} \sum_k \pi_{k|h} f_k(y_i)$$

(Asparouhov and Muthen 2008; Vermunt 2003). `group_membership` names cluster-level covariates of the group class. The group classes are returned by `get_results(fit, "group_classes")` and `"clusters"`.

Families

`"gaussian"` (identity link, a residual standard deviation per class, or one shared under `variance = "equal"`), `"binomial"` (logit link; the outcome is 0/1, logical, a two-level factor whose second level is the success, or `cbind(successes, failures)`), `"poisson"` (log link; `offset()` terms in the formula are honoured) and `"negative_binomial"` (log link, NB2: variance mu + alpha mu^2 with a dispersion alpha per class, or one shared under `variance = "equal"`, for overdispersed counts; a dispersion estimated at zero is the Poisson limit and raises `latents_boundary`).

`"ordinal"` fits a cumulative-logit (proportional-odds) regression within each class (McCullagh 1980),

$$P(Y \le c \mid x, k) = F(t_{kc} - x'\beta_k), \quad c = 1, \ldots, C-1,$$

with ordered thresholds $t_{k1} < \ldots < t_{k,C-1}$ per class in place of the intercept, and the parametrization of `MASS::polr()`: a positive slope increases the probability of higher categories. The outcome is an ordered factor, a factor (its levels taken in order) or whole-number categories; the categories are those observed, at least two. `common` terms share their slopes across classes, and `offset()` is honoured; the formula's intercept is replaced by the thresholds, so it must not be removed. The thresholds appear in the coefficient table as terms `"threshold:<lower>|<upper>"`, and `exp_estimate` of a slope is the cumulative odds ratio of a higher category. The class "mean" of an ordinal outcome (fitted values, `predict()`, trajectories) is the expected category score $\sum_c cP(Y = c)$, the categories scored 1 to C in order; `predict(type = "probabilities")` gives the category probabilities. With one outcome per class assignment, an ordinal mixture with no predictors is not

identified. Two-category mixtures follow the binary support restriction below. Both are rejected with `latents_not_identified`, unless `class_level = "group"` gives each class assignment several outcomes. A single-level mixture is also rejected when it has more free parameters than P (C - 1), the number of free category probabilities in P distinct covariate patterns (for example two classes and one binary factor).

Binary mixtures with one trial per class assignment are not supported: whether such a model is identified depends on the covariate design and the model constraints, which the package does not assess. A binary mixture therefore requires `class_level = "group"` (several rows per class assignment) or binomial counts with more than one trial per row; other binary specifications are rejected.

Conditions

`latents_bad_argument` for an inconsistent specification (including a `common` that names no predictor term); `latents_bad_data` for an outcome outside the family's support, a covariate that varies within a group where it may not, or a factor predictor with a single value among the fitted (positive-weight) rows; `latents_not_identified` for a binary outcome with one trial per class assignment, or an ordinal mixture with one outcome per class assignment and two categories, no predictors, or fewer distinct covariate patterns times C - 1 than free parameters; `latents_missing_data` for missing values under `missing = "error"`; `latents_rows_dropped` (warning) under `missing = "omit"`, and for rows of weight zero carrying a factor level or outcome category that no row of positive weight has; `latents_no_valid_start` when every start degenerated; `latents_unconverged` (warning) when the selected start did not converge; `latents_degenerate_start` (warning) when some starts degenerated; `latents_separation` (warning) for extreme fitted logits or log means that may signal separation; `latents_failed_starts` (warning) when some growth-mixture starts could not be built (they are marked `"failed"` in the starts table); `latents_boundary` (warning) for a negative-binomial dispersion at the Poisson limit, or an ordinal threshold gap at zero (a class that gives no row some middle category; that gap and the thresholds above it have no standard error); `latents_random_boundary` (warning) for a degenerate random-effect covariance; `latents_singular_information` (warning) when some standard errors cannot be computed.

References

Asparouhov, T., & Muthen, B. (2008). Multilevel mixture models. In G. R. Hancock & K. M. Samuelsen (Eds.), *Advances in latent variable mixture models* (pp. 27–51). Information Age.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42, 109–142.

Muthen, B., & Shedden, K. (1999). Finite mixture modeling with mixture outcomes using the EM algorithm. *Biometrics*, 55, 463–469.

Nagin, D. S. (2005). *Group-based modeling of development*. Harvard University Press.

Proust-Lima, C., Philipps, V., & Liquet, B. (2017). Estimation of extended mixed models using latent classes and latent processes: the R package lcmm. *Journal of Statistical Software*, 78(2), 1–56.

Verbeke, G., & Lesaffre, E. (1996). A linear mixed-effects model with heterogeneity in the random-effects population. *Journal of the American Statistical Association*, 91, 217–221.

DeSarbo, W. S., & Cron, W. L. (1988). A maximum likelihood methodology for clusterwise linear regression. *Journal of Classification*, 5, 249–282.

Wedel, M., & DeSarbo, W. S. (1995). A mixture likelihood approach for generalized linear models. *Journal of Classification*, 12, 21–55.

Follmann, D. A., & Lambert, D. (1991). Identifiability of finite mixtures of logistic regression models. *Journal of Statistical Planning and Inference*, 27, 375–381.

Vermunt, J. K. (2003). Multilevel latent class models. *Sociological Methodology*, 33, 213–239.

Leisch, F. (2004). FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8).

See Also

`enumerate_regressions()` to compare numbers of classes.

Examples

```
fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                          n_starts = 3, seed = 1)
fit
as.data.frame(fit)
get_results(fit, "classes")

# One class per student, several rows per student:
by_student <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                                 id = "student", class_level = "group",
                                 n_starts = 3, seed = 1)
get_results(by_student, "fit")


# Students nested in schools, each school in one of two group classes:
schools <- mixture_regression(score ~ wave, growth_schools, n_classes = 2,
                              id = "student", class_level = "group",
                              random = "wave", random_covariance = "equal",
                              cluster = "school", n_group_classes = 2,
                              n_starts = 2, seed = 1)
get_results(schools, "group_classes")
```

`multilca` *Fit a two-level latent class model*

Description

`multilca()` fits a latent class model for categorical data with observations nested within groups. It describes response patterns through latent classes and differences between groups through the relative prevalence of those classes. Groups with similar class compositions are represented by the same latent group class.

By default, the response probabilities defining the observation-level classes are shared across groups. The model estimates these probabilities together with class proportions and posterior membership probabilities, allowing both observations and groups to be classified while retaining information about classification uncertainty.

Usage

```
multilca(data, vars, id, n_profiles, n_group_classes = 2L, ...)
```

Arguments

data	A data frame with one row per observation.
vars	Names of the indicator columns. Numeric, integer, logical, character and factor columns are accepted; factor levels retain their declared order and other types are sorted.
id	Name of the group identifier column, or NULL for a single-level model.
n_profiles	Number of observation-level latent classes (profiles).
n_group_classes	Number of group-level latent classes. Defaults to 2, or to 1 when id = NULL.
...	Further arguments to multilpa(), such as n_starts, seed, missing, tol or profile_covariates. ordinal and count name ordinal and count indicators; the remaining indicators are categorical. categorical is accepted but has no further effect; its names must be in vars.

Details

The function calls multilpa(), treating all indicators in vars as categorical except those specified through ordinal or count in .... Categorical indicators have unrestricted class-specific response probabilities; ordinal and count indicators use the corresponding structured measurement models. For a combination of continuous and categorical indicators, use multilpa() with categorical.

Value

A fitted multilpa object (multilpa_covariates when membership covariates are specified), as returned by multilpa(). Categorical response probabilities are in get_results(fit, "responses"), one row per profile, item and category.

Conditions

latents_bad_categorical when categorical names a column not in vars; otherwise the conditions of multilpa().

See Also

multilpa() for continuous and mixed indicators, and vignette("lca", package = "latents").

Examples

```
activities <- c("time_with_friends", "on_social_media", "tv_video_games")
fit <- multilca(subset(student_esm, day <= 1), vars = activities,
                id = "student", n_profiles = 2, n_group_classes = 2,
                n_starts = 1, seed = 1)
get_results(fit, "responses")
get_results(fit, "profile_probabilities")
```

`multilpa` *Fit a two-level latent profile model*

Description

Multilevel latent profile analysis describes heterogeneity among observations and among the groups in which they are nested. Examples include repeated measurements within individuals and students within schools. Observation-level profiles describe patterns in the indicators, while group-level classes describe differences in the probabilities of those profiles.

In the standard model, profile definitions are shared across group classes: a given profile has the same indicator means and covariance structure in every group class. Group classes therefore differ in their profile composition. Observations within the same group can belong to different profiles, with their membership probabilities depending on the group's latent class.

`multilpa()` estimates the measurement parameters, conditional profile probabilities, group-class proportions, and posterior membership probabilities at both levels. The default specification uses Gaussian indicators with diagonal within-profile covariance matrices. Alternative specifications support other covariance structures, discrete or mixed indicators, and predictors of latent membership.

Usage

```
multilpa(
  data,
  vars,
  id,
  n_profiles,
  n_group_classes = 2L,
  profile_covariates = character(),
  group_covariates = character(),
  variance_model = c("varying", "equal"),
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  min_variance = NULL,
  seed = NULL,
  start = NULL,
  missing = c("error", "fiml"),
  profile_slopes = c("shared", "group_class"),
  covariance_model = c("diagonal", "full"),
  categorical = character(),
  min_probability = 1e-10,
  time = NULL,
  fixed = character(),
  centering = c("none", "person", "grand"),
  volume = NULL,
  shape = NULL,
```

```
  orientation = NULL,
  model = NULL,
  select_start = c("likelihood", "converged"),
  prior = NULL,
  noise = FALSE,
  acceleration = c("squarem", "none"),
  family = c("profiles", "additive", "dispersion", "additive_dispersion",
    "restricted_cross_level", "full_cross_level"),
  between_variance = c("varying", "equal"),
  weights = NULL,
  ordinal = character(),
  count = character(),
  count_model = c("poisson", "negative_binomial"),
  count_dispersion = c("varying", "equal"),
  scaling = c("none", "person", "grand"),
  drop_empty = TRUE,
  cores = 1L,
  local_dependence = character(),
  direct_effects = character(),
  search = NULL,
  n_screen = NULL,
  boundary = c("allow", "exclude")
)
```

Arguments

`data`	A data frame containing the indicators and the group identifier.
`vars`	Unique character vector of indicator column names. Columns not named in `categorical`, `ordinal` or `count` are continuous indicators.
`id`	Name of the group identifier column. Character, factor and numeric identifiers are supported; missing identifiers are not. `id` has no default: omitting it raises `latents_bad_argument`. `id = NULL` fits a **single-level** model: each row is its own unit, the observations are independent, and `n_group_classes` is one. The fit raises a `latents_single_level` message. Methods for extracting results, visualization and inference remain available. The internal unit column is named `.observation`; it is not returned by `get_results(x, "data")`, and a `data` that already has a column of that name raises `latents_bad_data`. Requesting more than one group class with `id = NULL` raises `latents_bad_argument`.
`n_profiles`	Positive integer number of individual profiles.
`n_group_classes`	Positive integer number of latent group classes (default 2). With `id = NULL` it is one, and any other value is an error.
`profile_covariates, group_covariates`	Names of numeric columns of `data` that predict individual-profile and group-class membership through multinomial logits, with the last class as reference. Naming either fits the one-step maximum-likelihood covariate model and returns a `multilpa_covariates` object. Profile intercepts differ by group class;

profile slopes are shared across group classes unless `profile_slopes = "group_class"`. A group covariate must be constant within each group. Covariates enter in their supplied units, without centring or scaling. The covariate model rejects `start` and `fixed`; it accepts `missing = "fiml"` for the indicators, but the covariates must be complete. `three_step()` and `r3step()` are the staged alternatives.

`variance_model` `"varying"` (the default; profile-specific variances or covariance matrices) or `"equal"` (shared across profiles).

`n_starts` Positive integer number of EM starts (default 10). The first start is deterministic: Ward's hierarchical clustering of the standardized indicators (on at most 2000 evenly spaced rows), each cluster initialized with its own means and variances, and, with more than one group class, Ward's clustering of the groups' profile compositions. The remaining starts are k-means from random centres. A supplied `start` replaces the first start, and the remaining starts are random. With `start` and `max_iter = 0`, exactly one start is evaluated regardless of `n_starts`. Zero-weight units do not contribute to initialization. For `search = "emEM"`, `n_starts` is the number of final optimization runs, including the protected first start (see `search`).

`max_iter` Nonnegative integer maximum number of EM updates per start (default 1000). `max_iter = 0` performs no update: with `start`, the model is evaluated at exactly those values, for example to score them with `logLik()`; without `start`, the random initializations are evaluated and the one with the highest log likelihood is returned.

`tol` Positive convergence tolerance (default `1e-8`). EM has converged when one step (or SQUAREM cycle) changes the log likelihood by no more than `tol` times the number of observed indicator values (each non-missing cell in a row of positive weight, with a minimum count of one). The tolerance is therefore expressed per observed indicator value.

`min_variance` Lower bound on each variance, or on each covariance eigenvalue under full covariance; the fit is then constrained maximum likelihood. `NULL` (the default) sets it to `1e-6` times the smallest sample variance of the continuous indicators (observed values in rows of positive weight, on the scale fitted, after any centring and scaling), so that the default bound adapts to the measurement scale. A numeric value specifies an absolute bound in squared indicator units. The bound used is stored as `min_variance` on the fit. Estimates at the bound are reported and raise a warning.

`seed` Optional random seed: any integer `set.seed()` accepts, including negative integers. When supplied, the caller's random-number state is restored on exit.

`start` Optional list of `means` and `variances` (profiles by indicators), `profile_probabilities` (group classes by profiles), and `group_probabilities` (vector). Starting probabilities must be positive. For full covariance, supply `covariances` (indicators by indicators by profiles); `variances` may then be omitted, or must match their diagonals. A categorical model also takes `response_probabilities`, a list of one profiles-by-categories matrix per indicator named in `categorical`. The list is matched by position unless it is labelled: when its elements are named after the indicators, or its columns after the categories, each block is matched to the indicator and category its labels name. A label naming an indicator or category the fit does not have raises `latents_bad_start`.

`missing` `"error"` (the default) rejects missing indicators; `"fiml"` maximizes the observed-data likelihood under an ignorable missingness mechanism (MAR), integrating missing indicators out rather than filling them in. This also holds with membership covariates; the covariates themselves must be complete (`latents_bad_data`).

`profile_slopes` `"shared"` (the default) gives each profile covariate one slope per profile logit, common to every group class. `"group_class"` estimates separate slopes in each group class (a cross-level interaction), reported with terms such as `z:group_class_1`; it multiplies the number of slope parameters by `n_group_classes`, each set estimated from the groups in its class. Naming it without `profile_covariates` raises `latents_bad_argument`.

`covariance_model` `"diagonal"` (the default) assumes conditional independence; `"full"` estimates within-profile residual covariances.

`categorical` Character vector naming indicators to treat as categorical. Each is modelled by unrestricted, profile-specific response probabilities over its observed categories (the latent class measurement model); binary, ordinal and unordered indicators are handled by the same parameterization. Numeric, integer, logical, character and factor columns are accepted. Indicators not named here stay Gaussian, so naming a subset fits a mixed-mode model.

`min_probability` Positive lower bound on every categorical response probability (default `1e-10`), constraining the maximum-likelihood problem as `min_variance` does for Gaussian indicators.

`time` Optional name of a column giving each observation's position within its group, such as a wave, occasion or course number. It is not used in estimation; it is stored for `get_results(x, "sequences")`, `get_results(x, "sequence_summary")` and `plot(what = "sequences")`. Values must be complete and unique within each group, and are ordered as `sort()` orders them: numbers numerically, a factor by its levels, and character strings alphabetically. Numbers stored as text whose alphabetical order is not their numeric order (`"2"`, `"10"`), and numbered labels sharing their text around the number whose alphabetical order is not the numbers' (`"T2"`, `"T10"`; `"week 2"`, `"week 10"`), raise `latents_bad_time` (also `latents_bad_data`); give the order with `factor(levels = )` or zero-pad the numbers. Zero-padded numbers and ISO dates are accepted.

`fixed` Character vector naming measurement blocks to hold at the values in `start` instead of estimating them: any of `"means"`, `"variances"` and `"response_probabilities"`, or `"measurement"` for every block the model has. Under `covariance_model = "full"`, `"variances"` holds the residual covariance matrices. A fixed block retains its supplied values in every start and is not counted in `n_parameters`, so the result is a different model, not a different starting point. `start` must contain the named blocks; `starting_values(fit, what = "measurement")` produces them, and `fit_staged()` performs the two-stage workflow in one call. The mixing parameters cannot be held.

`centering` How to centre the continuous indicators before fitting. `"none"` (the default) fits them as supplied. `"person"` subtracts each group's own mean from its rows, so values are deviations from the unit's average, the profiles describe within-unit change rather than level, and the group classes are types of change pattern rather

than types of unit (the person-mean-centred within-person design; Quintana, 2021; Voelkle, Brose, Schmiedek, & Lindenberger, 2014). `"grand"` subtracts one mean per indicator, which moves the origin without changing the within-group structure. The offsets are stored on the fit, so `get_results(x, "data")` returns the supplied columns and row-alignment checks still apply. `"person"` raises `latents_bad_data` when it leaves an indicator constant, as happens when a unit has one observation of it.

`volume`, `shape`, `orientation`

The covariance structure in its three components. Each profile's covariance decomposes as `Sigma_k = lambda_k * D_k * A_k * D_k'`, with *volume* `lambda_k = |Sigma_k|^(1/d)`, *orientation* `D_k` (eigenvectors) and *shape* `A_k` (diagonal, determinant one). Constraining the three across profiles gives the fourteen models mclust names with three letters, all of which are available.

`volume` is `"equal"` or `"varying"`. `shape` is `"equal"`, `"varying"` or `"spherical"`; `"spherical"` makes the variance equal across indicators within a profile and leaves no orientation to constrain. `orientation` is `"axis"` (a diagonal covariance), `"equal"` (one orientation shared by every profile) or `"varying"`. Each defaults to `NULL`, which follows `variance_model` and `covariance_model`.

volume	shape	orientation	model
equal	spherical	—	EII
varying	spherical	—	VII
equal/varying	equal/varying	axis	EEI, VEI, EVI, VVI
equal/varying	equal/varying	equal	EEE, VEE, EVE, VVE
equal/varying	equal/varying	varying	EEV, VEV, EVV, VVV

The estimates are those of Celeux and Govaert (1995); EVE and VVE, which combine a shared orientation with a free shape, have no closed form and use the minorize-maximize step of Browne and McNicholas (2014). Parameter counts match mclust's for all fourteen.

Structures other than EEI, VVI, EEE and VVV are maximized across all profiles jointly and cannot be combined with a held `variances` block. `parameter_inference()` reports Wald standard errors for all fourteen, computed in the structure's free coordinates (log volumes, determinant-one log shapes and orientations) and transformed to the reported variances and covariances by the delta method; `method = "bootstrap"` is also available for all fourteen.

`model`

The covariance structure as an mclust code, one of `"EII"`, `"VII"`, `"EEI"`, `"VEI"`, `"EVI"`, `"VVI"`, `"EEE"`, `"VEE"`, `"EVE"`, `"VVE"`, `"EEV"`, `"VEV"`, `"EVV"`, `"VVV"`. The letters give the volume, shape and orientation, each equal (`E`) or varying (`V`) across profiles, with `I` for axis-aligned (no covariances). These are the codes `enumerate_classes()` takes. Use it instead of `variance_model`/`covariance_model` or `volume`/`shape`/`orientation`, not together with them.

`select_start`

Rule for selecting a fitted solution. `"likelihood"` (the default) selects among starts within `tol * max(1, n_observed)` of the highest log likelihood, preferring converged starts within this set. Here `n_observed` counts observed indicator values in positive-weight rows. `"converged"` restricts selection to converged

starts when any are available; otherwise it uses all starts. Remaining ties are resolved by start order. Convergence is required for Wald inference but does not establish identification or global optimality. `summary()` and `get_results(fit, "starts")` report the individual starts. The reported replication count includes final-stage starts within `1e-6 * max(1, n_observed)` of the selected log likelihood, where `n_observed` counts observed indicator values in rows of positive sampling weight, as in the EM stopping rule. This absolute likelihood-gap tolerance is independent of measurement units and of `tol`, since a start stopped by the EM rule can still lie several steps below its maximum. It is stored as `replication_tolerance` and does not select a start.

`prior` `NULL` (the default) for maximum likelihood, or a `prior_control()` object for maximum a posteriori estimation of the Gaussian means and covariances under the conjugate prior of Fraley and Raftery (2007). Hyperparameters use the data-dependent defaults of `mclust::defaultPrior()`, and updates follow the conventions described in `prior_control()`. Available for EII, VII, EEI, VEI, EVI, VVI, EEE, EEV, VEV and VVV (mclust has no prior for VEE, EVE, VVE and EVV), with complete continuous indicators and without `fixed` or membership covariates; otherwise `latents_unsupported_prior` is raised. The mixing proportions have no prior. As in mclust, the fit's `log_likelihood` is the *unpenalized* log likelihood at the posterior mode, and `aic`, `bic` and their variants are computed from it with the usual parameter count. Convergence uses the change described under `tol`; because a MAP step can lower that likelihood, the decrease check made under maximum likelihood is not applied. `parameter_inference(method = "wald")` raises `latents_unsupported_inference` for such a fit, because the score is not zero at a posterior mode; `method = "bootstrap"` refits every resample with the same prior. The fit stores the request as `prior` and the resolved hyperparameters, in the indicators' units (centred, under `centering`), as `prior_parameters`.

`noise` `FALSE` (the default), or `TRUE` to add a noise component: a mixture component with constant density `1 / V` over the hypervolume `V` of the data, which absorbs outliers and scatter that no Gaussian profile explains (Banfield & Raftery, 1993; mclust's `initialization = list(noise = )` and `Vinv`). `V` is mclust's `hypvol()`: the smaller of the volumes of the data's axis-aligned and principal-component-aligned bounding boxes. The component has no measurement parameters; its mixing proportion is estimated with the others, and, as in mclust, it adds two parameters (the proportion and the hypervolume), following mclust's parameter-count convention. A random start assigns probability 0.1 to noise. A supplied `start` includes `noise_probability` alongside `profile_probabilities`, the two summing to one. Available for complete continuous indicators with one group class (`id = NULL` or `n_group_classes = 1`), under any of the fourteen covariance structures, with or without `prior`; otherwise `latents_unsupported_noise` is raised. In the fit the noise component is profile `0`, as in mclust's classification: `subject_profiles` is `0` for an observation assigned to it, `get_results(fit, "assignments")` adds a `posterior_noise` column, `get_results(fit, "posteriors")` and `get_results(fit, "profile_probabilities")` have profile-`0` rows, and the classification diagnostics count it as a class. `means`, `variances`, `covariances`, `profile_probabilities` and `subject_posteriors` describe the Gaussian profiles only, so the last two sum to one minus the noise share; the fit adds `noise_probability`,

noise_posteriors, n_noise and hypervolume. parameter_inference(), three_step(), r3step(), bootstrap_lrt(), starting_values(), bivariate residuals, fit_staged() and the posterior plots raise latents_unsupported_noise for such a fit.

acceleration "squarem" (the default) accelerates EM with SQUAREM (Varadhan & Roland, 2008): each cycle extrapolates along the last two EM steps in unit-free coordinates (means divided by reference standard deviations, variances and covariances by the corresponding products; probabilities unchanged). Reference standard deviations use observed values in rows of positive sampling weight. Each cycle ends with an ordinary EM step and reverts to the EM update when the extrapolated point would lower the likelihood, so the iteration stays monotone. Convergence is assessed over a complete cycle. "none" runs ordinary EM. iterations and max_iter count EM steps under either setting. Fits with prior, and models with membership covariates, always use plain EM. For EVE and VVE, whose shared orientation has no closed form, each EM iteration takes a fixed number of warm-started orientation steps (a generalized EM step) under either setting, which keeps every iteration monotone and the EM map a fixed function, as SQUAREM requires.

family The model family. "profiles" (the default) is the model described above: individual profiles whose prevalences differ across group classes. Three group-class-only families have no individual profiles (Houle, Morin & Harvey, 2026): each group has a Gaussian intercept per indicator, drawn from its group class's distribution, and its members vary around it.

- "additive": group classes differ in their means (and, optionally, in between-group variances); one within-group variance per indicator is shared by every class.
- "dispersion": group classes differ in their within-group variances; means and between-group variances are shared.
- "additive_dispersion": group classes differ in means and in within-group variances.

These families take only data, vars, id, n_group_classes, between_variance, n_starts, max_iter, tol, min_variance, seed, weights, search and n_screen; any other argument raises latents_bad_argument. Groups of sampling weight zero are not part of the sample: the identification checks, starting values, counts, entropy and classification tables use only the groups of positive weight, though every group still receives a posterior. Indicators must be complete and continuous, and at least one group must have more than one row (latents_unidentified otherwise). Between variances estimated at zero are reached by a boundary maximization with a Karush-Kuhn-Tucker check and flagged (latents_boundary); interior fits are finished by Newton steps. The result has class multilpa_additive; bic uses the number of groups and bic_individual the number of rows. It is read with get_results(); summary(), plot(), coef(), vcov(), confint() and parameter_inference() work on it, enumerate_classes() compares families through its family argument, and bootstrap_lrt() tests one number of group classes against another. These families are experimental: parameter_inference(method = "bootstrap") raises latents_unsupported_inference, and missing indicators (latents_bad_data), missing, membership covariates and ordinal or

count indicators (latents_bad_argument) are not available. See the group-class and cross-level models article.

Two cross-level families estimate individual profiles and group classes from the same indicators, following the manifest-aggregation specification of Houle et al. (2026, supplement): each group's means of the indicators are between-level indicators of its group class, with class-specific means and variances.

- "restricted_cross_level": profile prevalences are the same in every group class, so profiles and group classes are estimated separately; the profile composition of each class is reported descriptively.
- "full_cross_level": profile prevalences differ across group classes (as in "profiles"), and the group means also inform the classes.

These take data, vars, id, n_profiles, n_group_classes, variance_model (individual profiles), between_variance (group classes), n_starts, max_iter, tol, min_variance, seed and weights. The group means are computed from the same ratings, so the likelihood is the specification's working likelihood: comparable between cross-level fits of the same data, not with the other families. Standard errors are not available (latents_unsupported_inference). The result has class multilpa_cross_level, read with get_results(). Its get_results(fit, "profiles") table ends with a prevalence column: each individual profile's prevalence (estimated and shared for the restricted family, the class-weighted marginal for the full family) and, on the group-class rows, the class weight.

between_variance

For the additive and additive-dispersion families: "varying" (the default) estimates between-group variances per group class, "equal" one set shared by all classes. The dispersion family holds them equal and rejects "varying". For the cross-level families, the variances of the group means within each group class.

weights NULL, or the name of a numeric column of data holding a sampling weight for each independent unit: each id group of a two-level fit, each row of a single-level one. A two-level weight must be constant within its group (latents_bad_weights otherwise); within-unit weights are not supported. The fit maximizes the pseudo log likelihood sum_j w_j log L_j (Skinner, 1989) with the weights scaled to sum to the number of units of positive weight, as Mplus scales to the sample size, so integer weights give the fit to the data with each unit repeated that many times and the criteria stay on the sample's scale. A unit of weight zero is not part of the sample: it changes no estimate, likelihood, sample size, entropy or criterion, sets neither the grand offset nor the grand scale of centering/scaling = "grand", and is not classified. Nor does it define a category: a categorical, ordinal or paired (local_dependence) value seen only on units of weight zero is not a category of the fit and has no parameter, and an indicator constant among the positive-weight units is rejected regardless of zero-weight values. Under missing = "error", a missing categorical, ordinal or count value on a unit of weight zero is not rejected; a missing continuous value is. Standard errors are sandwich estimates: parameter_inference() defaults to vcov_type = "robust" and rejects "observed" and "opg" (latents_unsupported_weights), and method = "bootstrap" resamples the units with their weights. Classification tables report each unit's own posterior; effective counts and proportions

are weighted. bootstrap_lrt(), three_step(), r3step(), prior and noise reject a weighted fit. Weights are supported with every family, membership covariates, categorical indicators and missing = "fiml".

ordinal
: Names of indicators in vars to model as ordinal: an (ordered) factor, or integers, whose categories are ordered by factor level or by value. Each follows an adjacent-category logit with category intercepts shared by every profile and one location per profile, log P(k | c) / P(k - 1 | c) = a_k - a_(k-1) + eta_c, with the last profile's location fixed at zero: (K - 1) + (C - 1) parameters per indicator, against C (K - 1) for a categorical one. This is Latent GOLD's default ordinal model, against which it is checked. The category probabilities and locations are returned by get_results(fit, "ordinal"). A factor level never observed in the data raises the latents_unused_category warning; droplevels() removes it.

count
: Names of indicators in vars holding non-negative whole numbers, modelled as Poisson with one mean per profile (or negative binomial, see count_model); the means are returned by get_results(fit, "count_means"). Ordinal and count indicators support missing = "fiml", weights, two-level fits, membership covariates, Wald and bootstrap inference, bootstrap_lrt(), predict() and lta(). start, fixed, prior and noise do not accept them (latents_unsupported_indicator), and the group-class and cross-level families do not support them (latents_bad_argument).

count_model
: "poisson" (the default) or "negative_binomial": NB2, with mean mu and variance mu + alpha mu^2, for counts with greater variance than a Poisson within a profile (Latent GOLD's poisson overdispersed). A dispersion estimated at zero is the Poisson limit, a boundary fit (latents_boundary warning; Wald inference is rejected there).

count_dispersion
: For count_model = "negative_binomial": one dispersion per profile ("varying", the default) or one shared by every profile ("equal").

scaling
: How to scale the continuous indicators before fitting, at the same levels as centering. "none" (the default) fits them as supplied. "grand" divides each indicator by its standard deviation over all observations; "person" divides each group's values by that group's own standard deviation. With the matching centring these are z-scores: centering = "grand", scaling = "grand" standardizes each indicator over the sample, and centering = "person", scaling = "person" standardizes it within each unit. Standard deviations use the observed values with denominator n - 1. The scales are stored on the fit with the centring offsets, so get_results(x, "data") returns the supplied columns, and predict() scales new units by their own standard deviation under "person" and by the fitted standard deviation under "grand". Means, variances and covariances are reported in standard-deviation units. The log likelihood and every information criterion are those of the scaled values, so only fits with the same centring and scaling are comparable. A standard deviation that is zero or undefined (under "person", a unit observed once on an indicator or always giving the same value) raises latents_bad_data. Under EEI, VEI, EVI, VVI, EEE, VEE, EVV and VVV, whose family is unchanged when each indicator is rescaled, grand scaling changes only the units of the estimates and adds the density Jacobian to each start's log likelihood. SQUAREM uses unit-free coordinates, so its search path

is equivariant in exact arithmetic when variance bounds are inactive or transformed with the model. Rounding near convergence can still change iteration counts, and nearly tied starts can change the selected labelling. A scalar variance or eigenvalue floor is not generally preserved by unequal indicator rescaling. Under the spherical `EII` and `VII`, which impose one variance on every indicator, and under `EEV`, `VEV`, `EVE` and `VVE`, which share eigenvalues or eigenvectors across profiles (neither survives rescaling the indicators by different factors), it changes the solution. Under `centering = "person"` with `scaling = "person"`, a group with only two observations of an indicator becomes exactly `-1/sqrt(2)` and `1/sqrt(2)` on it: such groups raise `latents_atomic_groups`, and when every group is like that the fit is rejected with `latents_bad_data`.

`drop_empty` `TRUE` (the default) or `FALSE`: how the reported statistics treat rows whose every indicator is missing, and groups consisting only of such rows. This setting affects reporting, not estimation. With `TRUE` they are excluded, as in Mplus, from the individual and group sample sizes of the information criteria, from relative entropy, AvePP and the entropy term of ICL, AWE and CLC, from the classification tables and diagnostic plots, and from the effective class counts. Row-aligned assignment and posterior tables retain these units but report their classifications and posterior probabilities as `NA`. A partially observed row remains classified, and an empty row in an informative group can retain its group-class assignment. With `FALSE` every row and group is counted in those statistics, except in the individual sample size of a `multilpa()` fit, which excludes such rows either way. Units of sampling weight zero are excluded from every count and entropy term under either setting. `compare_models()` and `bootstrap_lrt()` do not compare fits made with different `drop_empty`. The setting is stored on the fit and passed on by enumerations and refits; it has no effect on data without such rows.

`cores` Number of processor cores for the random starts (default 1). The starting values are drawn in order in the current process and only the EM iterations are distributed, so the result is identical to `cores = 1`. Under `search = "emEM"` both the screening of the pool and the final runs are distributed; the screening is most of the time of an emEM fit, so a large model or data set gains most. Workers are forked processes on macOS and Linux, and a socket cluster on Windows and on macOS with Apple's Accelerate BLAS, as in `enumerate_classes()`. Lost worker results are retried once with half as many workers (rounded up). One `latents_worker_lost` warning reports losses and recovery; starts still lost are recorded as failed. Socket workers retain completed chunks, but tasks within a failed chunk may also be retried. Models with membership covariates run their starts on one core. To fit many models, give `cores` to `enumerate_classes()`, which shares its worker pool across candidates and their starts for models without membership covariates or joined categorical `local_dependence` pairs. Other fitting engines use one task per candidate.

`local_dependence`

Pairs of indicators that stay associated within each class, written `"item1:item2"` (several as a character vector). A continuous indicator pair has a covariance estimated within each class (one shared covariance with `variance_model = "equal"`), every other covariance staying zero: the covariance Latent GOLD fits with `item1 <-> item2 | Cluster`. It is not combined with `covariance_model`,

model or the structure arguments, covariates, time, prior, noise or held variances. A pair of categorical indicators is modelled jointly within each class. Each such pair becomes one categorical indicator, named "item1:item2", whose categories are every combination of the two items' categories (the full grid: an unobserved combination is estimated on the min_probability floor and counted as a parameter, so each class has K1 K2 - 1 of them), so every class has its own joint distribution of the pair: the class-specific association Latent GOLD fits with item1 <-> item2 | Cluster. The other indicators stay independent given the class. get_results(fit, "local_dependence") reports each class's joint probabilities beside their values under independence; predict(), parameter_inference(data =) and the other functions that take data join the pair in the caller's data. Under missing = "fiml" a row observing one item of a pair and not the other contributes the observed item, the pair's joint distribution summed over the missing one, which is the observed-data likelihood under MAR. A pair that mixes a categorical and a continuous indicator, that names an indicator twice or that shares an indicator with another pair raises latents_bad_argument; a categorical item of a pair whose categories contain ":" raises latents_bad_data. Default: none.

direct_effects Direct effects of covariates on continuous indicators, written "indicator ~ covariate" (several covariates joined with +, several indicators as a character vector). The covariate moves the indicator's mean by the same amount in every profile, y = mean_k + b z + e (Bandeen-Roche et al., 1997), so a difference in the indicator that the covariate explains is not read as a difference between profiles: the direct effect Latent GOLD fits with y <- 1 + Cluster + z. Covariates are numeric columns of data, observed in every row (code a categorical covariate as indicator columns), and enter with their values as given, so means are the profile means at a covariate value of zero; centre a covariate first to read them at its mean. The effects are estimated with the profiles (a conditional maximization step in EM; Meng & Rubin, 1993), have standard errors in parameter_inference() (parameter direct_effect, term "indicator:covariate"), and are reported by get_results(fit, "direct_effects"). predict() and the functions that take data read the covariates from it. Direct effects need a diagonal covariance (EEI or VVI) and are not combined with local_dependence between continuous indicators, profile_covariates, group_covariates, time, prior, noise, fixed, start, centering or scaling; those, a term that is not "indicator ~ covariate", and an indicator that is not continuous raise latents_bad_argument, and a covariate that is absent, not numeric, missing or constant raises latents_bad_data. Default: none.

search How the starts are found: NULL (the default), "kmeans" or "emEM". NULL uses "emEM" when every profile shares one covariance matrix (EII, EEI, EEE with continuous indicators), which removes the isolated-profile covariance-collapse mechanism. Shared covariances can still hit their floor on degenerate or over-extracted data; inspect the boundary diagnostics. The default is "kmeans" otherwise, including categorical-only models and the group-class families. The selected search strategy is recorded on the fit. "kmeans" runs n_starts starts to convergence: Ward's clustering first, then k-means from random centres. "emEM" screens a pool of n_screen starts by a short EM run each and runs n_starts selected starts to convergence (Biernacki, Celeux & Govaert, 2003).

The first start (Ward's clustering, or supplied starting values) is retained for final optimization if its screening succeeds; remaining starts are selected by screening likelihood. Thus `n_starts = 1` finalizes Ward's start whenever it screens successfully, even if another screened start has a higher likelihood. With a supplied start and `n_starts = 1`, screening is skipped entirely. The pool begins with the kmeans search's own starts, drawn in the same order, followed by random partitions of the rows and random rows as class centres, and `get_results(fit, "starts")` gives every start of the pool with its `kind`, its `stage` and its likelihood after screening. Either search reports the start with the highest likelihood among those run to convergence (subject to `select_start` and `boundary`). With profile-specific variances that maximum can correspond to a nearly degenerate component concentrated on a few observations (McLachlan & Peel, 2000, section 3.10). More extensive searches can identify such solutions more often. A variance at `min_variance` produces a `latents_boundary` warning; use `boundary = "exclude"` to exclude these starts when an interior solution is available. Low effective Gaussian profile membership produces the heuristic warning `latents_thin_profiles`. `get_results(fit, "gaussian_diagnostics")` reports continuous covariance and support measures without an automatic spike cutoff or changing the selected start. Neither a warning's absence nor a repeated likelihood establishes a regular or globally optimal solution.

`n_screen` For `search = "emEM"`, the number of starts screened, at least `n_starts`. `NULL` (the default) screens five times `n_starts`. The protected first start is included in the `n_starts` final optimization runs. Increasing `n_screen` does not increase that number. A single supplied start with `n_starts = 1` disables screening. A value with `search = "kmeans"`, or one below `n_starts`, raises `latents_bad_argument`.

`boundary` Whether a start with a variance at `min_variance` (a covariance eigenvalue, under full covariance) may be reported. `"allow"` (the default) permits it, as Mplus does, with a `latents_boundary` warning. `"exclude"` sets such starts aside and applies `select_start` to the rest; for models without covariates, the `boundary` column of the starts table identifies them. When every start is at the bound, the best is reported with a `latents_all_starts_boundary` warning. A Gaussian profile can increase the likelihood without limit by shrinking around one observation, so the size of that gain is set by `min_variance` rather than by the data, and more starts make such a solution more likely to be found. Categorical indicators have no variances and are unaffected.

Details

Infinite and constant observed indicators are rejected; each indicator must have at least two distinct observed values. No rows are silently dropped. Under FIML, fully missing individuals contribute no direct measurement likelihood but receive posterior probabilities from their group's information. The individual-level BIC excludes these rows through `n_informative`; the treatment of fully missing groups follows `drop_empty`. Missing patterns can prevent parameter identification; no general identification guarantee is made. Only initialization fills missing values with indicator means; EM uses conditional Gaussian sufficient statistics and observed marginal densities. More than one group class requires more than one profile and at least one group with more than one individual; these checks are necessary but do not establish identification. A model of complete categorical and ordinal indicators alone, with one group class, warns (`latents_unidentified`) when it has more

free parameters than its contingency table has free cells (negative degrees of freedom), which no data can identify (Goodman, 1974). Start selection follows `select_start`; the default can return a solution that has not converged. Inspect the `converged` flag and the `starts` table before interpreting the model. Profile and group-class labels are arbitrary.

Value

A `multilpa` object containing `means`, `variances`, optional `covariances` (indicators by indicators by profiles), `profile_probabilities`, `group_probabilities`, posterior matrices, classifications, log likelihood, information criteria, restart diagnostics and convergence history. Individual rows retain their input order; groups keep first-occurrence order. `bic` and `bic_groups` use the observed group count; `bic_individual` uses `n_informative`, the number of rows with at least one observed indicator. These are alternative conventions, not interchangeable criteria. `boundary` identifies variance bounds; `small_classes` flags a profile or group class whose effective membership (`effective_profile_counts`, `effective_group_counts`, weighted for a fit with `weights`) is below one, in which case the fit warns with `latents_small_classes`. The fit carries no standard errors: `parameter_inference()` computes them, and `bootstrap_lrt()` tests nested models. Global optimality is not guaranteed for any `n_starts`. `as.data.frame()` returns the tidy tables, selected by `what`.

References

Vermunt, J. K. (2003). Multilevel latent class models. Sociological Methodology, 33, 213–239. doi:10.1111/j.0081-1750.2003.t01-1-00131.x.

Goodman, L. A. (1974). Exploratory latent structure analysis using both identifiable and unidentifiable models. Biometrika, 61(2), 215–231. doi:10.1093/biomet/61.2.215.

Banfield, J. D., & Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. Biometrics, 49, 803–821. doi:10.2307/2532201.

Bandeen-Roche, K., Miglioretti, D. L., Zeger, S. L., & Rathouz, P. J. (1997). Latent variable regression for multiple discrete outcomes. Journal of the American Statistical Association, 92(440), 1375–1386.

Meng, X.-L., & Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. Biometrika, 80(2), 267–278. doi:10.1093/biomet/80.2.267.

Houle, S. A., Morin, A. J. S., & Harvey, J.-F. (2026). Multilevel latent profile analyses: A comprehensive guide. Organizational Research Methods. doi:10.1177/10944281261469432.

Skinner, C. J. (1989). Domain means, regression and multivariate analysis. In C. J. Skinner, D. Holt, & T. M. F. Smith (Eds.), Analysis of complex surveys (pp. 59–87). Wiley.

Asparouhov, T. (2005). Sampling weights in latent variable modeling. Structural Equation Modeling, 12, 411–434. doi:10.1207/s15328007sem1203_4.

Fraley, C., & Raftery, A. E. (2007). Bayesian regularization for normal mixture estimation and model-based clustering. Journal of Classification, 24, 155–181. doi:10.1007/s00357-007-0004-5.

See Also

`multilca()` for a model in which every indicator is categorical.

Examples

```
# Course sessions nested in students: engagement profiles of sessions, and
# classes of students that differ in how often they are engaged.
activity <- c("browse", "lectures", "forum_read", "forum_post", "attendance")
fit <- multilpa(course_engagement, activity, "student",
                n_profiles = 2, n_group_classes = 2, n_starts = 4,
                seed = 42)
summary(fit)
as.data.frame(fit)
get_results(fit, what = "profile_probabilities")
```

`parameter_inference` *Standard errors, confidence intervals and tests for model parameters*

Description

`parameter_inference()` quantifies sampling uncertainty in the parameter estimates of a fitted mixture model. It provides standard errors and confidence intervals for quantities such as indicator means, response probabilities, membership coefficients, and transition probabilities, according to the model family. These results describe the precision of the estimates and support interpretation of the fitted model.

Two approaches are available for supported models. Wald inference uses an estimated parameter covariance matrix to obtain standard errors, confidence intervals, and tests for eligible parameters. Bootstrap inference repeatedly resamples independent observational units and refits the model, using variation across the resulting estimates to calculate standard errors and percentile confidence intervals.

The function returns a table of parameter estimates and their uncertainty measures. The available parameters, covariance estimators, and inference methods depend on the fitted model. Wald tests are omitted where an ordinary test against zero is inappropriate, including probabilities and variance parameters. Bootstrap results contain standard errors and confidence intervals but no hypothesis-test p-values.

Usage

```
parameter_inference(
  x,
  data = NULL,
  level = 0.95,
  step = 1e-04,
  vcov_type = c("observed", "robust", "opg"),
  adjust = .multilpa_p_adjust_methods,
  method = c("wald", "bootstrap"),
  iter = 199L,
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  seed = NULL,
```

```
  boundary = c("error", "fix")
)
```

Arguments

x	A converged fit from multilpa(), lpa(), lca() or multilca() (with or without covariates), lta(), the additive family, or a growth mixture. A fit with an active bound raises latents_boundary_fit and an unconverged fit latents_no_converge. Mixture regressions raise latents_bad_argument (their inference is in get_results(), confint() and vcov()), and the cross-level families raise latents_unsupported_inference.
data	Optional data frame the model was fitted to. When omitted it is rebuilt from the indicators, identifiers and occasions the fit stores. A supplied frame is checked exactly against the stored indicators and identifiers; for an older fit without stored indicators, the likelihood gives a weaker consistency check.
level	Confidence level, strictly between zero and one. Default 0.95.
step	Positive finite-difference step for the analytic score derivative. Default 1e-4.
vcov_type	Covariance estimator for the Wald method. "observed" (default) inverts the observed information. "robust" returns the Huber-White sandwich A^-1 B A^-1, where B is the outer product of the per-group scores, multiplied by the CR1 small-sample factor G / (G - 1) for G groups (the default of sandwich::vcovCL(), applied to every robust standard error in the package). Mplus's MLR applies no such factor, so these errors are sqrt(G / (G - 1)) times Mplus's. Groups are the independent units: robust errors relax the assumption that the within-group model is correctly specified, not the assumption that groups are independent. "opg" inverts the outer product of the per-group scores (the BHHH estimator). The OPG and observed-information estimators are asymptotically equivalent under a correctly specified regular model, but can differ in finite samples. "robust" and "opg" need more groups than parameters; "opg" also needs group scores of full rank, since it inverts their outer product, while the sandwich does not. For a fit with sampling weights the default is "robust", and any other value raises latents_unsupported_weights. For a fit with centering = "person" or scaling = "person" the default is also "robust", and "observed" or "opg" raise latents_unsupported_inference. Transforming observations within a unit induces dependence; the sandwich estimator aggregates score contributions at the independent-unit level. Under centering = "grand" or scaling = "grand" the sample means and standard deviations the indicators were transformed with are estimated jointly with the model, as estimating equations, and their variability is added to every covariance type: a centred mean mu_k - xbar has the standard error of that difference, not that of mu_k. vcov_type and step are not used by method = "bootstrap".
adjust	Multiplicity correction applied to p_value to give p_adjusted: "none" (default) or one of the stats::p.adjust() methods. The method applied is stored in the adjust attribute. The correction is taken over the tests the table reports; bounded parameters carry no test and are not counted.
method	"wald" (default) inverts the observed information in the estimation coordinates; it is available for all fourteen covariance structures. EEI, VVI, EEE and VVV are estimated in log variances or log-Cholesky coordinates; the ten structures

that constrain volume, shape or orientation across profiles are estimated in log volumes, log shapes with the determinant-one constraint built into the coordinates, and Cayley-transform orientations (or, for VEE and EVV, determinant-one log-Cholesky factors), so the information has one row per free parameter. The reported variances and covariances are obtained from these coordinates by the delta method. Constraints can make the natural-scale covariance singular; the sum constraints on class probabilities also imply a singular covariance. `"bootstrap"` resamples groups with replacement, refits within the same covariance structure, realigns each refit's labels, and reports the percentile interval and the standard deviation of the replicates; it is also available for all fourteen structures and assumes independent groups, as `vcov_type = "robust"` does.

`iter` Number of resamples for `method = "bootstrap"`, at least two. Default `199`; a 95% percentile interval is more stable with 999 or more.

`n_starts`, `max_iter`, `tol`
Passed to each bootstrap refit, as in `multilpa()`.

`seed` Optional seed for the resampling: a whole number accepted by `set.seed()`; any other value raises `latents_bad_argument` with either method. The Wald method uses no random numbers. A supplied integer seed restores the caller's `.Random.seed` on exit; `NULL` uses its current stream. R does not expose the Box-Muller cached normal variate, so that cache cannot be restored; see base::Random.

`boundary` Handling of categorical response probabilities at their lower bound, `min_probability`, where the information is singular. `"error"` (default) raises `latents_boundary_fit`. `"fix"` holds those probabilities at the bound and reports every other parameter conditional on them, as Mplus does; the held probabilities are omitted from the table (or, for a covariate fit, reported with `NA` standard errors) and named in the `fixed_at_bound` attribute.

Details

For a covariate fit (`multilpa(profile_covariates = )`), the table covers every free parameter at all three levels: the measurement model, the profile logits and the group-class logits. `outcome` is the profile or group class a coefficient predicts, or the profile a measurement parameter belongs to, and `parameter` is `"mean"`, `"variance"`, `"covariance"`, `"response"` or `"coefficient"`. Each categorical indicator contributes one `"response"` row per profile, indicator and category, named `indicator:category`: the category probability, with its standard error from the multinomial logits and no Wald test. Class-membership coefficients are on the multinomial logit scale, with the final profile and the final group class as references, so each coefficient is a log odds against that reference. The tests are Wald tests, which are invalid for a coefficient driven to the boundary by separation or an empty class. A fit with a membership logit above 20 in coordinates independent of the covariates' units and origin (roughly, a slope per standard deviation of its covariate and an intercept at the covariates' means) raises `latents_boundary_fit`; the fit itself warns with `latents_extreme_coefficients`. Only `method = "wald"` is implemented for a covariate fit; `"bootstrap"` raises `latents_unsupported_inference`.

For a latent transition model from `lta()`, the table covers the measurement model, the group-class shares, each group class's initial profile distribution and its transition matrix. An initial probability (`parameter = "initial_probability"`, `level = "profile"`) has `outcome` the profile and `term` the group class; a transition probability (`"transition_probability"`) has `outcome` the profile

moved to and term group_class_h:profile_j, the group class and the profile moved from. The scores follow the Fisher identity and the observed information is the numerical Jacobian of that analytic score; vcov_type = "robust" clusters on the groups, which are the sequences. method = "bootstrap" resamples persons, refits and matches each replicate's profiles and group classes to the original's; it needs data (latents_bad_argument otherwise). A transition row with no information (a profile no group occupies before its last occasion) raises latents_boundary_fit.

For a growth mixture (mixture_regression() with random), the table has one row per natural parameter: block ("coefficient", "membership" and, for a multilevel growth mixture, "group_membership", "residual_sd", "random_variance", "random_sd", "random_correlation", "share"), class ("all" for a quantity shared by every class), term, estimate, standard_error, statistic, p_value, p_adjusted, conf_low and conf_high. Wald rows use the fit's covariance (vcov_type) with the delta method; intervals of standard deviations and variances are formed on the log scale, of correlations on the Fisher scale and of shares on the logit scale, and only coefficients and logits are tested against zero. method = "bootstrap" resamples persons, refits each resample from n_starts starts and matches its classes to the original by their posterior memberships of the original persons; it is unavailable for a multilevel growth mixture (latents_unsupported_inference). Weak class separation can reduce the accuracy of Wald intervals; bootstrap inference provides an alternative assessment of sampling uncertainty. A class whose random-effect covariance is degenerate, or (with correlated residuals) a residual standard deviation or residual process held on its bound, raises latents_boundary_fit under boundary = "error", naming which; under "fix" those rows have no standard error or interval. data, when given, must be the fitted data; step is not used (the information is the derivative of the analytic score).

For a general transition fit (an lta() fit with covariate, occasion-varying or second-order transitions), the table has one row per transition, initial and second-order logit: block, group_class, from, to, term, estimate, standard_error, statistic, p_value, p_adjusted, conf_low, conf_high, estimated (and previous for a second-order fit). estimated is FALSE for a logit the likelihood does not identify: a move into an occasion at which no row observes an indicator, under transitions = "occasion" (or every second-order logit when nothing is observed from occasion 3 on). Such a logit is not counted in n_parameters, AIC or BIC, and its standard error and test are NA. method = "bootstrap" resamples persons, refits, and matches every replicate's profiles (on the measurement) and group classes to the original's, rebasing the initial logits when the reference profile moves; it needs data, and works where Wald inference is refused (for example full covariance structures).

Transition inference (both kinds of lta() fit) fixes no parameter at a bound: without an active bound boundary = "fix" gives the same table as "error", and a fit with an active bound raises latents_boundary_fit under either.

Value

A base data.frame with one row per reported parameter and the columns level ("measurement", "profile" or "group"), outcome, term, parameter, estimate, standard_error, statistic, p_value, p_adjusted, conf_low and conf_high. Estimates are on the natural scale (variances in squared indicator units, probabilities on [0, 1]). A parameter whose null value lies on the boundary of its space (a variance, a probability, a covariance diagonal, a count mean) has NA statistic, p_value and p_adjusted, but still has an interval. All class probabilities are reported; their sum constraints make the natural covariance singular. Wald intervals for probabilities are formed on the logit scale and for variances (and covariance diagonals) and count means on the log scale, then back-transformed, so they stay within the parameter's range; every other interval is

estimate +/- z * standard_error. confint() returns the same intervals. A probability at or below 1e-6 is treated as on its bound (refused, or held by boundary = "fix").

A fit made with fixed reports only the parameters it estimated: a held measurement block contributes no row to the table or to the information matrix. The table has one row per estimated natural coefficient: n_parameters rows plus one for each set of probabilities whose reference category the estimation scale drops. Standard errors, intervals and p-values are conditional on the held measurement solution and do not include its uncertainty (see fit_staged()). For a fit_staged() result, method = "bootstrap" refits both stages in each resample, so the table includes the measurement parameters and that uncertainty is propagated (attribute stages_resampled is 2).

With method = "bootstrap" the table has the same columns and rows; estimate is the fitted value, standard_error the standard deviation of the replicates and conf_low/conf_high the percentile interval. statistic, p_value and p_adjusted are NA. Its attributes are covariance (the covariance of the replicates, returned by vcov()), level, method, iter, n_valid, replicates (the kept replicates, one row each), messages (why a resample was dropped, NA where it was not), structure and fixed; covariance_unconstrained is NULL.

For the Wald method, diagnostics of the whole fit are attributes: covariance and covariance_unconstrained (natural and estimation-scale covariance of the estimates), hessian, gradient, scaled_score, score_displacement (the largest score times its estimation-scale standard error, which the latents_unconverged warning uses), condition_ratio, level, step, method, vcov_type, fixed (the measurement blocks the fit held, character() when none), fixed_at_bound (the probabilities held by boundary = "fix") and adjust. When vcov_type is "robust" or "opg", group_scores holds the groups-by-parameters score matrix and scaling_correction the MLR scaling correction factor tr(A^-1 B) / q (with B without the CR1 factor), used for scaled likelihood-ratio difference tests; it is NA otherwise.

Interpretation

Inference is conditional on the specified model, including its number of classes and measurement assumptions. The reported uncertainty does not account for selecting that specification from several candidates. Wald inference requires an adequately converged and locally identified solution; estimates near parameter boundaries may require special treatment. Bootstrap inference also depends on successful refitting and consistent matching of class labels across resamples.

Computation

Numerical differentiation of the score requires two score evaluations per free parameter. For profile, class, covariate and transition models, the information matrix is cached within an R session for the same fitted model, data and step. Subsequent inference, plotting and get_results() calls reuse this calculation; changes to its inputs trigger recalculation.

Conditions

Errors: latents_no_converge for an unconverged fit; latents_boundary_fit when a variance, response probability or mixing probability is at its bound; latents_singular_information when the observed information cannot be inverted; latents_no_free_parameters when fixed held every parameter; latents_bad_inference_data when a supplied data does not reproduce the fit; latents_too_few_groups for vcov_type = "robust" or "opg" when the fit has no more independent groups than free parameters. The Wald method raises latents_unsupported_inference

for a fit made with `multilpa(prior = )` (a posterior mode, where the score is not zero), and either method raises `latents_unsupported_noise` for a fit with a noise component. A fit whose score is still far from zero gives a `latents_unconverged` warning. An invalid `level`, `step`, `iter`, `n_starts`, `max_iter` or `tol` raises `latents_bad_argument`, and a `data` that is not a data frame `latents_bad_data`. A fitted mixture regression raises `latents_bad_argument` naming `get_results(x, "coefficients")`, which reports its standard errors and tests; an object that is not a fitted model raises `latents_not_a_fit`.

`method = "bootstrap"` also raises `latents_unsupported_inference` for an additive or covariate fit, and for a fit holding a measurement block through `multilpa(fixed = )`, since resampling the data does not resample the held values (a `fit_staged()` result is bootstrapped by refitting both stages instead), and `latents_bootstrap_failed` when fewer than two resamples gave a usable fit; its message gives the first reason. Resamples that fail or do not converge are dropped with a `latents_bootstrap_dropped` warning stating how many.

Examples

```
set.seed(42)
example_data <- data.frame(
  school = rep(seq_len(10), each = 10),
  score_a = rnorm(100), score_b = rnorm(100)
)
fit <- multilpa(example_data, c("score_a", "score_b"), "school",
                n_profiles = 2, n_group_classes = 1, n_starts = 2, seed = 1)
parameter_inference(fit)

# Apply Benjamini-Hochberg adjustment to the reported tests.
parameter_inference(fit, adjust = "BH")

# Wald inference for a VEI model with equal shape across profiles.
shaped <- multilpa(example_data, c("score_a", "score_b"), "school",
                   n_profiles = 2, n_group_classes = 1, n_starts = 2, seed = 1,
                   volume = "varying", shape = "equal", orientation = "axis")
parameter_inference(shaped)

# Resample schools and refit the same covariance structure.
parameter_inference(shaped, method = "bootstrap", iter = 10, n_starts = 1,
                    seed = 1)

# A covariate fit: membership driven by `x` and by the group's class.
set.seed(5)
school <- rep(seq_len(16), each = 8)
high_class <- rep(rep(c(FALSE, TRUE), length.out = 16), each = 8)
x <- rnorm(128)
profile <- ifelse(
  runif(128) < plogis(-1 + 2 * high_class + 0.8 * x), 2L, 1L
)
covariate_data <- data.frame(
  school = school, x = x,
  y1 = rnorm(128, ifelse(profile == 2L, 2, -2), 0.7),
  y2 = rnorm(128, ifelse(profile == 2L, 1.5, -1.5), 0.7)
)
```

```
covariate_fit <- multilpa(covariate_data, c("y1", "y2"), "school",
                          n_profiles = 2, n_group_classes = 2,
                          profile_covariates = "x", n_starts = 2, seed = 1)
parameter_inference(covariate_fit, covariate_data)

# Inference for a latent transition model, including transition probabilities.
moves <- lta(subset(course_engagement, student <= 40),
             c("browse", "lectures", "forum_read"), "student",
             n_profiles = 2, time = "sequence", n_starts = 2, seed = 1)
parameter_inference(moves)
```

`pool_imputations` *Fit and pool mixture models across multiple imputations*

Description

`pool_imputations()` estimates a mixture model from multiply imputed data and combines the results to account for uncertainty associated with missing values. It fits the same model specification to each completed data set, allowing parameter estimates to reflect the alternative plausible values supplied by the imputation procedure.

Class labels can differ across fitted mixture models even when their profiles are similar. Before pooling, the function matches individual profiles and group classes to those in the first imputation's fit. It then combines the aligned parameter estimates and covariance matrices using Rubin's rules. The resulting uncertainty incorporates both the estimated sampling variance within each analysis and variation in estimates between imputations.

The function returns pooled estimates, standard errors, confidence intervals, and tests for eligible parameters, together with summaries of the individual analyses. Completed data sets must be supplied as a list or a `mids` object from mice; the function does not construct the imputation model.

Usage

```
pool_imputations(
  imputed,
  vars,
  id,
  n_profiles,
  ...,
  level = 0.95,
  vcov_type = c("observed", "robust", "opg"),
  adjust = .multilpa_p_adjust_methods
)
```

Arguments

`imputed`	The completed data sets: a list of at least two data frames with the same columns and rows in the same order, or a `mids` object from the mice package, which is completed here.

vars, id, n_profiles
: As in multilpa().

...
: Further arguments for multilpa(), such as n_group_classes, profile_covariates, group_covariates, categorical, n_starts and seed, used for every imputation.

level
: Confidence level of the pooled intervals (default 0.95).

vcov_type
: "observed" (default), "robust" or "opg", as in parameter_inference(), applied to every imputation's fit. For fits with sampling weights the default is "robust", and the other types are refused with latents_unsupported_weights; for fits with person centring or scaling the default is "robust" too, and the other types raise latents_unsupported_inference.

adjust
: Multiplicity correction applied to the pooled p-values to give p_adjusted, as in parameter_inference(): "none" (default) or a method of stats::p.adjust(). The per-imputation tables are computed without correction.

Details

Pooling is on the reported (natural) scale: the pooled estimate is the mean over imputations, and its variance is the mean within-imputation variance plus (1 + 1/m) times the between-imputation variance. Degrees of freedom follow Rubin (1987) with a large complete-data sample. Pooled intervals are symmetric t intervals, rather than the transformed Wald intervals used for bounded parameters by parameter_inference(). A structural constant, such as the probability of one for a single class, has zero within- and between-imputation variance and zero missing information. When the within-imputation variance is zero and the between-imputation variance is positive, riv is infinite and fmi is one.

Value

An object of class latents_pooled. as.data.frame() and get_results(x, "estimates") give one row per parameter, with the columns of parameter_inference() (level, outcome, term, parameter, estimate, standard_error, statistic, p_value, p_adjusted, conf_low, conf_high) plus df (Rubin's degrees of freedom), within and between (the two variance components), riv (the relative increase in variance due to missingness) and fmi (the fraction of missing information). statistic and p_value are NA where parameter_inference() reports no test (variances and probabilities). get_results(x, "imputations") gives each imputation's aligned estimates, one row per imputation and parameter, and get_results(x, "fits") one row per imputation with its log likelihood, convergence and the label permutation that aligned it.

Interpretation

Pooling requires an imputation model appropriate for the analysis and missingness assumptions, including the grouping structure where relevant. The number of classes and other model specifications remain fixed across imputations. Label matching resolves differences in numbering but cannot establish that substantially different class solutions represent the same underlying profiles.

Estimates are pooled on their reported scale. Consequently, symmetric confidence intervals for probabilities or variances can extend beyond their permissible ranges.

Choosing the imputation model

Valid pooling requires an imputation model compatible with the analysis model and the assumed missingness mechanism. Associations between covariates and indicators may be nonlinear because they depend on latent class membership. The imputation model should represent these associations, account for clustering, include relevant indicators and predictors of missingness, and propagate uncertainty in its estimated parameters. Imputing separately within modally assigned classes omits classification uncertainty. Complete-case analysis can also change the within-class indicator distributions when missingness depends on the indicators.

Conditions

`latents_bad_data` when `imputed` is not a list of at least two data frames with identical columns and row counts; `latents_pooling_failed` when any imputation's fit or inference fails (the message names the imputation and gives the original reason); `latents_missing_package` for a `mids` object without mice installed. Warnings from the individual fits are passed on as they are raised.

References

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

Meng, X.-L. (1994). Multiple-imputation inferences with uncongenial sources of input. *Statistical Science*, 9(4), 538–558.

van Buuren, S. (2018). *Flexible Imputation of Missing Data* (2nd ed.), section 5.2. Chapman & Hall/CRC.

Examples

```
set.seed(1)
# Illustrative completed data sets for demonstrating the pooling interface.
# For an analysis, supply data generated by an appropriate imputation model.
completed <- lapply(seq_len(2), function(i) {
  within(subset(course_engagement, student <= 40),
         previous_grade <- previous_grade + rnorm(length(previous_grade), sd = 0.1))
})
pooled <- pool_imputations(completed, c("browse", "lectures", "forum_read"),
                           "student", n_profiles = 2, n_group_classes = 1,
                           profile_covariates = "previous_grade",
                           n_starts = 2, seed = 1)
pooled
get_results(pooled, "fits")
```

---

predict.latents_mixture_regression

Predict from and simulate a fitted model

Description

`predict()` applies a fitted model to the original sample or new observations to obtain class membership probabilities or predicted outcomes.

For latent profile and class models, it estimates each observation's probability of belonging to each class, given its observed indicators, and identifies the most probable class. These probabilities express uncertainty in classification; the assigned class provides a discrete summary of that uncertainty.

For mixture regression models, it predicts outcomes from the relationships estimated within each class, either separately by class or averaged over class membership probabilities. Growth mixture models also provide individual trajectories informed by the person's observed outcomes.

`simulate()` generates replicate outcomes under a fitted mixture regression or growth mixture model, preserving the original predictors and grouping structure. The simulations reflect the model's assumed variation in class membership and outcomes while holding the estimated parameters fixed.

Usage

```
## S3 method for class 'latents_mixture_regression'
predict(
  object,
  newdata = NULL,
  type = c("response", "class_response", "posterior", "probabilities"),
  ...,
  trials = NULL
)

## S3 method for class 'latents_mixture_regression'
simulate(object, nsim = 1, seed = NULL, ...)

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

Arguments

`object`	A fitted model.
`newdata`	`NULL` for the fitted rows, or a data frame with the indicator or predictor columns (and `id` and membership covariates where the model uses them).
`type`	For a profile or class model: `"class"` (modal profile and group class with their posteriors), `"posterior"` (every profile's probability as well) or `"density"` (each row's log density, marginal over group classes, for scoring or held-out likelihood). For a mixture regression: `"response"` (the mean outcome over classes; expected successes for a binomial count response), `"class_response"` (one row per observation and class), `"posterior"` (class probabilities given the outcome, which `newdata` must contain) or `"probabilities"` (each category under each class; ordinal family only). For a growth mixture: `"response"` (the class trajectories averaged with each person's prior class probabilities), `"class_response"` (one row per observation and class, with random effects

set to zero), and, using outcomes in newdata, "posterior" (one row per person), "class_conditional" (one row per observation and class: the class trajectory plus the person's random effects predicted under that class, E(b | y, class)), "conditional" (those averaged with the posterior class probabilities) and "random_effects" (one row per person and class with its posterior probability; their posterior-weighted mean is lcmm::predictRE()).

`...` Unused.

`trials` For binomial mixture-regression predictions, the number of trials: one non-negative whole number, a vector with one value per row, or a column name in the prediction data. NULL uses the fitted trials when newdata = NULL, or infers them from the response columns when present. A count response otherwise requires trials; a Bernoulli fit defaults to one trial. For type = "posterior", any explicit value must agree with the observed response. Predictions remain on the expected success-count scale; trials = 1 requests probabilities for each trial.

`nsim` Number of simulated outcome vectors.

`seed` NULL uses and advances the caller's random stream. An integer makes simulation reproducible and restores the caller's .Random.seed. R does not store the cached Box-Muller normal draw in .Random.seed, so the next normal draw is not preserved when that generator has a cached draw.

Details

For multilpa() and multilca() models without membership covariates, it evaluates the E-step at the estimated parameters using the fitted categorical levels, centering, and missing-data rules. Applying it to the estimation data reproduces the fitted posterior probabilities for classified observations. In a two-level model, groups in newdata are treated as new groups and classified using their supplied observations. With the default drop_empty = TRUE, rows with every indicator missing retain their position but have NA for the predicted profile and its probabilities. A group is unclassified only when none of its rows has an observed indicator. Fitted rows with zero sampling weight are unclassified under either setting of drop_empty. New-data prediction does not apply sampling weights. These exclusions apply to classification output; density predictions retain each row's evaluated log density. For a model with a noise component, prediction supports complete rows and, under missing = "fiml", rows with every indicator missing. Partially observed rows are unsupported because the fitted noise hypervolume does not specify their marginal density. An entirely missing row has log density zero, including under the noise component.

For growth mixtures (mixture_regression(random =)), individuals and clusters in newdata are treated as new units. Class mean trajectories require only predictors; posterior class probabilities, individual trajectories, and empirical Bayes random effects also require observed outcomes. Prediction is not supported for membership-covariate or latent transition fits and raises latents_unsupported_prediction. For these fits, use get_results(x, "posteriors") to extract posterior probabilities for the estimation data.

simulate() draws new outcomes on the fitted design of a mixture regression or growth mixture: class memberships from the fitted mixing model (group classes first in a two-level model), random effects from each class's covariance, then outcomes from each row's class regression.

Value

A base data.frame with one row per row of newdata (per row and class for "class_response" and "class_conditional", per row, class and category for "probabilities"), starting with row, its position; "posterior" and "random_effects" of a growth mixture have one row per person (and class), starting with the id column. simulate() returns one row per fitted row and one column per simulation, sim_1, sim_2, ...

Conditions

latents_bad_data when newdata lacks an indicator or the id column, contains a category absent from the estimation data, contains unsupported missing values, or does not allow evaluation of the regression design. Binomial count predictions without trial counts also raise latents_bad_data; latents_bad_argument for "probabilities" outside the ordinal family; latents_unsupported_prediction for predict() on a membership-covariate fit or a latent transition fit. latents_unsupported_noise for partially observed prediction rows in a model with a noise component. A growth mixture prediction raises latents_bad_data when newdata lacks id, the cluster column or (for the outcome-based types) the outcome, and warns with latents_unused_argument for arguments it ignores.

Examples

```
fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                          n_starts = 2, seed = 1)
predict(fit, data.frame(hours = c(2, 10)), type = "class_response")
head(simulate(fit, nsim = 2, seed = 1))

# Predict an individual trajectory conditional on observed scores.
growth <- mixture_regression(score ~ wave, data = subset(growth_scores, student <= 60),
                             n_classes = 2, id = "student", class_level = "group",
                             random = ~ 1, n_starts = 1, seed = 1)
predict(growth, subset(growth_scores, student == 61), type = "conditional")
```

---

prior_control *Specify prior settings for Gaussian profile estimation*

Description

prior_control() specifies prior settings that regularise the estimation of Gaussian profile means and covariance parameters. The returned object is supplied to the prior argument of multilpa(). The prior applies to the continuous-indicator measurement parameters; class-membership probabilities are estimated without a prior.

The settings control mean shrinkage and the prior scale and degrees of freedom for covariance estimation. Unspecified values are determined from the data during fitting. The parameterisation follows the conventions of mclust, with effects that depend on the chosen covariance structure.

Usage

```
prior_control(shrinkage = 0.01, mean = NULL, dof = NULL, scale = NULL)
```

Arguments

shrinkage	Non-negative prior precision factor for the means, kappa (default 0.01, as in mclust). Zero leaves the means unshrunk.
mean	Optional prior mean, one value per continuous indicator on the scale used to fit the profiles. Use original indicator units when no transformation is requested, and the transformed scale under multilpa()'s centering or scaling. The separate internal centring used for numerical stability is handled automatically. Requires a positive shrinkage.
dof	Optional prior degrees of freedom, a single positive number.
scale	Optional prior scale: a single positive number for an axis-parallel or spherical structure, or a positive-definite d by d matrix for an ellipsoidal one.

Details

The prior follows Fraley and Raftery (2007) and mclust::priorControl(). Arguments set to NULL use the data-dependent defaults of mclust::defaultPrior(): the prior mean is the vector of indicator means, the degrees of freedom are d + 2 for d indicators, and the scale is (1/G)^(2/d) times the sample covariance (ellipsoidal structures) or times the average sample variance (axis-parallel and spherical structures), for G profiles.

The updates follow mclust's fitted-model convention, including its structure-specific departures from a textbook conjugate update, so the controls have different effects across covariance structures. For EII, scale does not enter the variance update. For EEI, shrinkage and mean enter the variance calculation but do not shrink the fitted means. For VEV, dof does not enter the update. EEI, EVI and EEV count the mean-prior contribution once rather than once per profile.

Value

An object of class latents_prior: a list with shrinkage, mean, dof and scale, the last three NULL where the default is to be computed from the data. Its print() method shows these settings and returns the object invisibly.

Interpretation

Using a prior changes the estimation procedure from maximum likelihood to regularised point estimation. It does not provide posterior samples or Bayesian credible intervals. Ordinary Wald inference is unavailable for these fits; supported bootstrap inference refits the model with the prior settings.

Conditions

latents_bad_argument for a negative or non-finite shrinkage, a non-finite mean, a mean with a zero shrinkage, a non-positive or non-finite dof, or a scale that is neither a positive number nor a square, symmetric, positive-definite matrix.

References

Fraley, C., & Raftery, A. E. (2007). Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of Classification*, 24, 155–181. doi:10.1007/s0035700700045

See Also

multilpa() for the supported model specifications.

Examples

```
prior_control()
prior_control(shrinkage = 0)
```

---

r3step *Relate latent class membership to external covariates*

Description

r3step() examines how external covariates are associated with membership in latent classes identified by a previously fitted model. It addresses whether the relative likelihood of belonging to particular classes varies with characteristics that were not used to define them. The measurement parameters remain fixed, preserving the class definitions while estimating their associations with the covariates.

Because class membership is unobserved, a regression that treats assigned classes as known outcomes can distort these associations. The three-step approach incorporates estimated classification-error probabilities into the membership regression, treating each assignment as an imperfect indicator of latent membership. Covariates enter a multinomial logistic model whose coefficients describe changes in the log odds of membership in a class relative to a reference class, holding the other covariates constant.

For supported multilevel models, the regression can concern individual profiles or group classes. For supported transition models, a separate specification relates covariates to initial profile membership and movement between profiles over time. The function reports regression coefficients, standard errors, confidence intervals, and hypothesis tests, with options for clustered inference and adjustment for multiple comparisons.

Usage

```
r3step(
  x,
  data,
  covariates = character(),
  level = c("individuals", "groups"),
  ci_level = 0.95,
  vcov_type = c("robust", "observed"),
  adjust = c("BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "none"),
  by_group_class = FALSE,
  initial_covariates = character(),
  transition_covariates = character()
)
```

Arguments

x	A covariate-free fit from multilpa() (family = "profiles", the default) or from lta() with its default homogeneous first-order transitions and invariant measurement, or a trajectory model from mixture_regression() that classifies persons (with or without random), whose person classes are then predicted. Fits with membership covariates and multilevel growth mixtures (cluster) raise latents_unsupported_three_step: the correction uses a fixed, unconditional classification-error matrix and independent units. The other families of multilpa(), general transition fits of lta() and mixture regressions with observation classes raise latents_unsupported_three_step too, except that a covariate-free general transition fit takes initial_covariates and transition_covariates.
data	The data frame carrying the covariates, in the fit's row order. Columns shared with the fit are checked row by row; a frame with too little identifying information raises a warning that alignment cannot be verified.
covariates	Character vector of numeric columns, without missing or infinite values, that are not measurement indicators of x. For level = "groups" each must be constant within a group; for a trajectory model each must be constant within a person and not part of the trajectory regression (latents_bad_covariate).
level	"individuals" (default) predicts profile membership, "groups" group-class membership. For a trajectory model the rows have level = "persons" and outcomes class_1, class_2, ...; "groups" is refused there with latents_bad_argument.
ci_level	Confidence level for the intervals (default 0.95).
vcov_type	"robust" (default) uses the sandwich clustered on the fit's groups with the small-sample (CR1) factor G / (G - 1) for G groups, as the default of three_step(). "observed" uses the observed information; for the pooled regression it treats every observation as independent, and with by_group_class = TRUE it is the model-based variance of the two-level likelihood. "robust" requires more independent groups than the regression has coefficients, because the group score contributions sum to zero at the estimate; otherwise it is refused with latents_too_few_groups, and "observed" remains available.
adjust	Multiplicity correction across the covariate terms of every non-reference class (not the intercepts), passed to stats::p.adjust(): "BH" (default), "holm", "hochberg", "hommel", "bonferroni", "BY" or "none".
by_group_class	FALSE (default) fits one pooled multinomial logit with a single intercept per profile. TRUE fits the two-level form of the one-step model: each group class has its own profile intercepts, the covariate slopes are shared across group classes, and a group's observations share one latent group class whose proportions are estimated jointly. Group classes represent between-group differences in profile membership that are not explained by the covariates. This changes the regression specification relative to the pooled analysis. Available only for level = "individuals" on a covariate-free multilpa() fit; refused with latents_bad_argument otherwise.
initial_covariates, transition_covariates	For an lta() fit with one group class: covariates of the profile at the first occasion (constant within a person) and of every transition (read at the destination occasion, so they may vary over occasions). Either one fits the three-step latent

Markov model described below instead of the occasion-by-occasion profile regression; covariates must then be empty, and level and by_group_class do not apply. Refused with latents_bad_argument on a fit that is not a transition model.

Details

On an lta() fit, level = "individuals" models the marginal profile at each occasion; it does not regress initial-state or transition probabilities on the covariates. vcov_type = "robust" clusters the occasion scores by group; "observed" treats them as independent. On a trajectory model the units are persons, each with one covariate value, and "robust" is the sandwich over persons.

The three-step latent Markov model (initial_covariates, transition_covariates) classifies each occasion from its own indicators, using the transition fit's measurement model and its profile shares pooled over occasions (the latent class model of the pooled data), and assigns it to its modal profile. Step three is a hidden Markov model for those assignments, with the classification errors fixed at their step-two values and the initial and transition logits regressed on the covariates (Di Mari, Oberski and Vermunt 2016). The transition fit's own posteriors are not used for the assignments, because they draw on a person's other occasions and their errors are therefore not independent across occasions, as step three assumes. The step-one fit must have no covariates and one group class (latents_unsupported_three_step otherwise). "robust" uses a sandwich covariance estimator with persons as independent units; "observed" uses the observed information.

Because uncertainty in the estimated classification-error matrix is omitted, interval coverage may be below the nominal level when class separation is weak. Inspect the estimated error matrix with get_results(x, "classification_errors").

Value

A base data.frame with one row per non-reference class and term, and the columns level, outcome, term, estimate, standard_error, statistic, p_value, p_adjusted, conf_low and conf_high. p_value is uncorrected; p_adjusted applies adjust (also recorded in the adjust attribute) and is NA on the intercept rows. A coefficient with zero standard error has NA_real_ statistic and p-value. Coefficients are log odds against the last class, as in multilpa(profile_covariates =); the reference class is named in the reference_class attribute and the variance used in the vcov_type attribute. With by_group_class = TRUE there is one intercept row per group class (group_class_1, group_class_2, ...), followed by the shared slopes and, at level = "groups", the group-class logits against the last group class; the result also carries the attributes by_group_class and log_likelihood (the maximised log likelihood).

With initial_covariates or transition_covariates, one row per coefficient of the step-three chain: block ("initial" or "transition"), from (the origin profile of a transition, NA for an initial logit), to, term, estimate, standard_error, statistic, p_value, p_adjusted, conf_low and conf_high, with the attributes log_likelihood, classification_errors (the step-two error matrix, rows the true profile), vcov_type and adjust. Initial logits are against the last profile and transition logits against staying, as in lta(initial_covariates =, transition_covariates =); p_adjusted covers the covariate terms.

Interpretation

The correction assumes that classification-error probabilities remain constant across covariate values within each latent class. Reported standard errors and confidence intervals treat those probabilities as fixed and omit uncertainty from their estimation and from selecting the class solution.

In the ordinary membership regression, the last class is the reference. A positive coefficient indicates higher odds of membership in the specified class relative to that reference; exponentiating the coefficient gives the corresponding odds ratio. Transition regressions instead compare movement to another profile with remaining in the origin profile. These associations do not, by themselves, establish causal relationships.

References

Vermunt, J. K. (2010). Latent class modeling with covariates: two improved three-step approaches. *Political Analysis*, 18, 450–469. Asparouhov, T., & Muthen, B. (2014). Auxiliary variables in mixture modeling: three-step approaches using Mplus. *Structural Equation Modeling*, 21, 329–341. Di Mari, R., Oberski, D. L., & Vermunt, J. K. (2016). Bias-adjusted three-step latent Markov modeling with covariates. *Structural Equation Modeling*, 23, 649–660.

See Also

`three_step()` for a distal outcome, and `multilpa(profile_covariates = )` for the one-step alternative that estimates everything jointly.

Examples

```
set.seed(21)
g <- rep(seq_len(50), each = 12)
x <- rnorm(600)
truth <- 1L + as.integer(runif(600) < plogis(-0.3 + 1.2 * x))
example_data <- data.frame(
  g = g, x = x,
  a = rnorm(600, ifelse(truth == 2L, 1.2, -1.2)),
  b = rnorm(600, ifelse(truth == 2L, 1.2, -1.2))
)
fit <- multilpa(example_data, c("a", "b"), "g", n_profiles = 2,
                n_group_classes = 1, n_starts = 4, seed = 1)
r3step(fit, example_data, "x")

# Three-step regression of transition probabilities on covariates.
moves <- lta(course_engagement,
             vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
             id = "student", n_profiles = 2, time = "sequence", n_starts = 2, seed = 1)
r3step(moves, course_engagement, transition_covariates = "sequence")
```

report *Display model estimates and diagnostic summaries*

Description

`report()` provides a combined overview of a fitted profile, class, or supported transition model. It brings together parameter estimates, descriptive statistics, and diagnostic checks to support examination of the estimated class structure, numerical convergence, classification uncertainty, and residual associations.

Output is displayed in the R session, with optional plots appropriate to the fitted model. The function returns the original fitted model invisibly. Individual tables remain available through `get_results()`, `descriptives()`, and `diagnostics()` for further analysis or reporting.

Usage

```
report(
  x,
  data = NULL,
  plots = TRUE,
  by = c("profile", "overall"),
  rows = 10L,
  ...
)
```

Arguments

`x`	A profile, class or transition fit: from `lpa()`, `lca()`, `multilpa()` or `multilca()` (with or without membership covariates), or from `lta()` with time-homogeneous transitions. A fit of another family (a mixture regression, growth mixture, general transition, additive or cross-level model) raises `latents_not_a_fit`; its own `summary()`, `diagnostics()` and `plot()` report it.
`data`	Optional data frame used to fit the model. The required columns are stored in the fitted object.
`plots`	`TRUE` (the default) draws the plots. `FALSE` prints only.
`by`	Passed to `diagnostics()` for the bivariate residuals: `"profile"` (the default) assesses each profile separately, `"overall"` pools them.
`rows`	Number of rows of each summary table to print, passed to `print(summary(x))`. Default 10.
`...`	Not used. Any further argument raises `latents_bad_argument` before anything is printed.

Value

The fitted model, invisibly. Called for its printed and plotted output.

Report contents

Output is presented in the following order: summary(), descriptives(), the checks from diagnostics(), and the supported plot views. Any omitted views are listed after the plots. The underlying tables can be accessed separately using get_results(), summary(), descriptives() and diagnostics().

See Also

summary(), diagnostics(), descriptives(), plot_views().

Examples

```
fit <- multilpa(
  course_engagement,
  vars = c("browse", "lectures", "forum_read", "forum_post", "attendance"),
  id = "student", n_profiles = 2, n_group_classes = 2, n_starts = 4,
  seed = 1
)
report(fit, plots = FALSE)
```

---

sensitivity *Assess sensitivity of a fitted mixture model to initialisation*

Description

sensitivity() evaluates how consistently a fitted mixture model is recovered from different random initialisations. Mixture-model estimation can converge to different local maxima, yielding different class assignments for the same data and model specification. Examining this dependence on initialisation helps assess the numerical reliability of the solution being interpreted.

The function repeats estimation under several random seeds and compares the resulting likelihood values and class assignments with the reference fit. Class labels are aligned before assignments are compared. By considering likelihood and classification agreement together, the analysis shows whether repeated searches recover a comparable fit and whether they assign the same observations to the same classes.

Results include convergence status, log likelihood, and classification agreement for each refit. Similar likelihood values are grouped using a specified tolerance. The data and model specification remain fixed throughout the analysis.

Usage

```
sensitivity(
  x,
  data = NULL,
  seeds = 1:10,
  n_starts = NULL,
  max_iter = NULL,
  tol = NULL,
  tolerance = NULL,
```

```
  cores = 1L
)
```

Arguments

x	The reference fit: a `multilpa()` model, an `lta()` model, or a growth mixture (`mixture_regression()` with `random`). A `fit_staged()` result varies the second-stage starts while holding its first-stage measurement solution fixed. A directly fitted model with measurement blocks held fixed is refused (`latents_unsupported_sensitivit` because its original free starting values are not retained. A membership-covariate fit raises `latents_unsupported_sensitivity`. For an `lta()` fit, `agreement` is the share of occasions given the same profile, after matching the refit's profiles on the measurement; for a growth mixture it is the share of persons given the same class, after matching the refit's classes by their posterior memberships of the fitted persons.
data	Optional for a `multilpa()` fit or a growth mixture, which carry the columns they were built from. Required for an `lta()` fit (`latents_bad_argument` otherwise). It must be the data the model was fitted to, in its original row order; shared identifier and indicator columns are checked against the fit (`latents_bad_inference_data`).
seeds	At least two distinct integer seeds accepted by `set.seed()`. Default `1:10`. Invalid values raise `latents_bad_argument`. Reusing the reference seed reproduces its initialisation when the other fitting controls are unchanged.
n_starts	Starts per refit. Defaults to the number the reference fit used.
max_iter, tol	Passed to each refit. Default to the reference fit's (for a fit that does not record them, to the defaults of the function that fitted it).
tolerance	Maximum log-likelihood difference for grouping refits. Each group is anchored at its highest log likelihood and spans no more than `tolerance`. Defaults to the reference fit's replication tolerance, 1e-6 per observed value.
cores	Number of processor cores (default 1). The seeds are refitted in parallel, one seed per worker at a time, each refit on one core. Every refit sets its own seed, so the result is identical to `cores = 1`. Workers are forked processes on macOS and Linux, and a socket cluster on Windows and on macOS with Apple's Accelerate BLAS, as in `multilpa()`; socket workers load the installed copy of the package. More cores than seeds are not used.

Details

Replication across seeds complements replication across starts within one fit. Each refit uses the reference fit's estimation controls unless alternatives are supplied.

A seed whose refit fails gives a row with `NA` estimates, and a `latents_sensitivity_dropped` warning reports how many seeds failed.

Value

A `data.frame` of class `latents_sensitivity` (which `plot()` draws), one row per seed, with columns `seed`, `log_likelihood`, `converged`, `iterations`, `optimum` (1 for the group containing the highest attained log likelihood, 2 for the next lower group, and so on), `best` (whether this seed

reached optimum 1) and `agreement` (the proportion of classified observations given the same profile as the reference fit, after label matching). For profile and transition models, this uses observations included in both fits under their `drop_empty` settings. A failed refit has `NA` in every column except `seed`. The `optimum` and `best` columns group attained likelihood values without filtering on convergence; inspect `converged` before interpreting them as evidence of replicated maxima.

`as.data.frame()` returns a plain `data.frame`. `plot()` accepts `main` and `subtitle` and returns a ggplot object showing each refit's log-likelihood difference from the best converged refit and its classification agreement with the reference fit. Unfinished refits are distinguished from converged maxima; if none converged, the highest observed log likelihood is the reference instead.

Interpretation

Consistent results across converged refits support numerical reproducibility, but do not guarantee a global maximum or establish the validity of the model. Similar likelihood values alone do not imply identical parameter estimates. The analysis assesses sensitivity to initialisation, rather than uncertainty arising from sampling or model selection.

References

Hipp, J. R., & Bauer, D. J. (2006). Local solutions in the estimation of growth mixture models. *Psychological Methods, 11*, 36–53.

See Also

`enumerate_classes()` to vary the number of classes, and `parameter_inference()` for uncertainty about the parameters of a single solution.

Examples

```
set.seed(7)
example_data <- data.frame(
  school = rep(seq_len(12), each = 10),
  score_a = c(rnorm(60, -1.5), rnorm(60, 1.5)),
  score_b = c(rnorm(60, -1), rnorm(60, 1))
)
fit <- multilpa(example_data, c("score_a", "score_b"), "school",
                n_profiles = 2, n_group_classes = 1, n_starts = 3, seed = 1)
sensitivity(fit, seeds = 1:3, n_starts = 3)

# Growth mixtures are known for local solutions (Hipp and Bauer 2006).
growth <- mixture_regression(score ~ wave, data = subset(growth_scores, student <= 60),
                             n_classes = 2, id = "student", class_level = "group",
                             random = ~ 1, n_starts = 1, seed = 1)
sensitivity(growth, seeds = 1:2)
```

`srl` *Self-regulated learning scores generated by a large language model*

Description

`srl` contains self-regulated learning scores derived from questionnaire responses generated by a large language model (GPT). Each row represents one of 300 synthetic respondents and contains five construct scores from the Motivated Strategies for Learning Questionnaire. Scores are means of the corresponding items, expressed on a scale from 1 to 7.

The dataset illustrates latent profile analysis with correlated indicators. Strong associations among four constructs provide an example for examining how assumptions about within-profile covariance influence fitted solutions. Test anxiety shows a different pattern of association with the remaining constructs.

Usage

```
srl
```

Format

A data frame with 300 rows and 5 columns:

`cognitive_strategies` cognitive strategy use

`intrinsic_value` intrinsic value of the course

`self_efficacy` self-efficacy

`self_regulation` self-regulation

`test_anxiety` test anxiety

Details

Correlations among cognitive strategies, intrinsic value, self-efficacy and self-regulation range from approximately 0.78 to 0.86. Their correlations with test anxiety range from approximately -0.30 to -0.10.

Source

The `SRL_GPT` data of the psychnets package, with descriptive column names. Saqr, M. (2025). Delving into the psychology of machines: Exploring the structure of self-regulated learning via LLM-generated survey responses. *Computers in Human Behavior*, 173, 108769. doi:10.1016/j.chb.2025.108769

Examples

```
descriptives(srl, names(srl))
```

`starting_values` *Extract starting values for a subsequent fit*

Description

Extract parameter values from a fitted model for use as starting values in a subsequent `multilpa()` fit. Measurement parameters can be extracted separately when membership probabilities should be re-estimated. Supplying starting values does not hold parameters fixed; use the fitting function's `fixed` argument to impose those constraints.

Usage

```
starting_values(
  x,
  covariance = c("auto", "drop", "keep"),
  what = c("all", "measurement")
)
```

Arguments

`x`	A fitted `multilpa` model, or a list carrying `profile_probabilities`, `group_probabilities`, and the measurement blocks the model uses: `means` with `variances` or `covariances` for continuous indicators, `response_probabilities` for categorical ones. An all-categorical model needs no Gaussian block. Other elements are ignored. A missing block raises `latents_bad_start`; a fit with a noise component raises `latents_unsupported_noise`; anything that is not a list raises `latents_bad_argument`.
`covariance`	`"auto"` (the default) keeps `covariances` when the object carries them, `"drop"` always returns the diagonal parameterization, and `"keep"` requires `covariances` and raises `latents_bad_start` when they are absent.
`what`	`"all"` (the default) returns the measurement and mixing blocks. `"measurement"` omits `profile_probabilities` and `group_probabilities`, which are sized for the model that produced them; use it with `fixed` in `multilpa()` for a staged fit that changes the number of group classes.

Details

This is an advanced utility for controlled refitting and selective parameter constraints. Required blocks are checked during extraction; further numerical and model-specific validation occurs during fitting.

Value

An object of class `multilpa_start`: the list `multilpa()` accepts as `start`, with elements `means`, `variances`, `profile_probabilities` and `group_probabilities`. When covariances are returned, `covariances` replaces `variances`; when the object carries a categorical measurement model, `response_probabilities` is added. `what = "measurement"` drops `profile_probabilities` and `group_probabilities`. The Gaussian blocks are unnamed and indexed by position; `as.data.frame()` returns them as

a labelled table. `response_probabilities` keeps each indicator's name and each block's category names, and `multilpa()` matches the blocks by these labels, so a fit whose categorical indicators or categories do not match raises `latents_bad_start`. The object can be passed directly as `start`.

See Also

`get_results()` for the values as tidy tables; `fit_staged()` for two-stage estimation with fixed measurement parameters.

Examples

```
set.seed(7)
example_data <- data.frame(
  school = rep(seq_len(12), each = 10),
  score_a = rnorm(120), score_b = rnorm(120)
)
fit <- multilpa(example_data, c("score_a", "score_b"), "school",
                  n_profiles = 2, n_group_classes = 1, n_starts = 2, seed = 1)
starting_values(fit)
as.data.frame(starting_values(fit))
refit <- multilpa(example_data, c("score_a", "score_b"), "school",
                    n_profiles = 2, n_group_classes = 1, n_starts = 1,
                    start = starting_values(fit))
logLik(refit)
```

`student_esm` *Leisure activities of university students in daily life*

Description

`student_esm` contains experience-sampling reports of leisure activities and affect from 100 university students. Each row represents a survey response collected when the student reported no studying since the previous prompt. Eight binary items describe activities undertaken during that interval, and four ratings describe affect.

Repeated reports are nested within students, providing an example for multilevel latent class analysis of activity patterns and differences in their prevalence between students. The dataset retains complete reports from students with at least 15 eligible responses. The affect ratings can also be included as indicators in a mixed measurement model.

Usage

```
student_esm
```

Format

A data frame with 2582 rows (100 students) and 15 columns:

student Integer student identifier, 1 to 100. The nesting unit; pass it as `id`.

day Integer study day, 0 to 13.

beep Integer prompt of the day, 1 to 5 among the prompts kept.

time_with_friends Factor, `"no"` or `"yes"`: spent time with friends since the previous prompt.

on_social_media Factor, `"no"` or `"yes"`: used social media.

tv_video_games Factor, `"no"` or `"yes"`: watched TV or played video games.

listened_music Factor, `"no"` or `"yes"`: listened to music.

sports Factor, `"no"` or `"yes"`: did sports.

walking Factor, `"no"` or `"yes"`: went for a walk.

reading Factor, `"no"` or `"yes"`: read.

part_time_job Factor, `"no"` or `"yes"`: worked in a part-time job.

happy Integer rating of feeling happy, 1 to 7.

relaxed Integer rating of feeling relaxed, 1 to 7.

worried Integer rating of feeling worried, 1 to 7.

exhausted Integer rating of feeling exhausted, 1 to 7.

Details

In the original study students answered six prompts a day for fourteen days; the activity items were asked only at prompts where the student had not studied since the previous prompt. The dataset contains 100 students, drawn with a fixed seed from those with at least 15 complete non-study reports, retaining every such report: 15 to 51 per student (median 24). The eight activity items are categorical indicators; the four affect ratings can be added as continuous indicators in a mixed measurement model. `worried` has a pronounced floor effect (49.7% of ratings are 1). A Gaussian component representing low-worry prompts may therefore reach the `min_variance` bound.

Source

openESM dataset 0062, deposited by Neubauer and Schmiedek at Zenodo, doi:10.5281/zenodo.17347974, under the Creative Commons Attribution 4.0 licence (CC-BY 4.0), and retrieved through the openESM database (`https://openesmdata.org`). The subset is produced by `data-raw/student-esm.R` in the source repository; identifiers are renumbered and the values are otherwise unchanged.

References

Neubauer, A. B., & Schmiedek, F. (2024). Approaching academic adjustment on multiple time scales. *Zeitschrift für Erziehungswissenschaft*, 27, 147–168. doi:10.1007/s11618023011828

See Also

`multilpa()` with `categorical =` for the two-level latent class model.

Examples

```
head(student_esm)

activities <- c("time_with_friends", "on_social_media", "tv_video_games",
                "listened_music", "sports", "walking", "reading",
                "part_time_job")
first_week <- subset(student_esm, day <= 6)
lca <- multilpa(first_week, vars = activities, id = "student",
                n_profiles = 2, n_group_classes = 2,
                categorical = activities, n_starts = 3, seed = 1)
get_results(lca, what = "responses")
get_results(lca, what = "profile_probabilities")
```

---

study_hours *Study hours and quiz scores under two study strategies*

Description

study_hours is a simulated dataset illustrating how the relationship between study time and quiz performance can differ across latent classes. Each row represents one study week for a student, with six weeks recorded for each of 150 students.

The data were generated from two week-level regression classes, labelled "deep" and "surface", with different intercepts and slopes relating study hours to quiz scores. Two student-level classes differ in the prevalence of these weekly classes, allowing class membership to vary across weeks within students.

The generating labels are retained for assessing class recovery. Additional binary and count outcomes provide examples of mixture regression with other response distributions.

Usage

```
study_hours
```

Format

A data frame with 900 rows (150 students by 6 weeks) and 10 columns:

student Integer student identifier, 1 to 150.

week Integer study week, 1 to 6.

hours Hours studied that week, 0 to 12.

sleep Hours slept the night before the quiz.

motivation Standardized motivation, constant within student.

score Quiz score.

passed Integer 0/1: passed the weekly check.

questions Count of questions the student asked that week.

strategy Factor, `"deep"` or `"surface"`: the week's generating strategy; retained for assessing class recovery.

student_type Factor, `"steady"` or `"erratic"`: the student's generating class; retained for assessing class recovery.

Details

The generating model contains two week-level study strategies with different regressions of quiz `score` on `hours`. The `"deep"` strategy has an intercept of approximately 35 and a slope of 4.5 points per hour; the `"surface"` strategy has an intercept of approximately 55 and a slope below one point per hour. Students in the `"steady"` class use the deep strategy in approximately 85% of weeks, compared with 25% for the `"erratic"` class. In the simulation, `motivation` predicts student-level membership and `sleep` predicts week-level membership.

`score ~ hours` is therefore a two-class mixture at the week level (`class_level = "observation"`) with class shares that differ by student (`id = "student"`, `n_group_classes = 2`). The binary outcome `passed` depends on the student's generating class and illustrates a mixture at the student level (`class_level = "group"`). `questions` is a count whose Poisson regression depends on the week's strategy.

Source

Simulated by `data-raw/study-hours.R` in the source repository.

See Also

`mixture_regression()`, `enumerate_regressions()`.

Examples

```
fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                          n_starts = 3, seed = 1)
get_results(fit, "coefficients")
```

---

three_step *Compare latent classes on an external outcome*

Description

`three_step()` estimates differences in an external outcome between latent classes defined by a previously fitted model. It addresses whether classes distinguished by their indicator patterns also differ on a variable that was not used to identify them. The measurement model remains fixed, so the outcome comparison retains the original meaning of the classes.

Latent class membership is inferred rather than observed. Assigning each observation to its most probable class and comparing the resulting groups treats those assignments as certain, which can distort estimated outcome differences. Three-step methods address this problem by using classification-error probabilities from the fitted model when estimating class-specific outcome means. The default

Bolck–Croon–Hagenaars (`"bch"`) method applies correction weights; the alternative maximum-likelihood (`"ml"`) method incorporates the classification-error probabilities into a model for the outcome.

The analysis provides estimated outcome means within classes or differences between pairs of classes, together with standard errors and confidence intervals. Pairwise differences can also be tested against zero, with adjustment for multiple comparisons. For supported multilevel models, comparisons may concern individual profiles or group classes. Uncorrected modal-assignment and posterior-weighted analyses are available as additional methods.

Usage

```
three_step(
  x,
  data,
  outcome,
  level = c("individuals", "groups"),
  method = c("bch", "proportional", "modal", "ml"),
  ci_level = 0.95,
  contrast = c("none", "pairs"),
  adjust = c("BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "none"),
  vcov_type = c("cluster", "independent"),
  variance = c("varying", "equal")
)
```

Arguments

`x`	A covariate-free fit from `multilpa()` (`family = "profiles"`, the default) or from `lta()` with its default homogeneous first-order transitions and invariant measurement, or a trajectory model that classifies persons (`mixture_regression()` with `class_level = "group"`, with or without `random` effects) fitted without `membership` covariates. Fits with membership covariates and multilevel growth mixtures (`cluster`) raise `latents_unsupported_three_step`: the correction uses a fixed, unconditional classification-error matrix and independent units. The other families of `multilpa()`, `lta()` fits from the general transition engine (occasion-varying, second-order or covariate transitions, occasion-specific measurement, a stayer class, ordinal or count indicators) and mixture regressions with observation classes raise `latents_unsupported_three_step` too.
`data`	The data frame carrying the outcome, in the fit's row order. Columns shared with the fit are checked row by row; a frame with too little identifying information raises a warning that alignment cannot be verified. For a trajectory model it must be the data frame the model was fitted to.
`outcome`	Name of a numeric outcome column, without missing or infinite values, that is not a measurement indicator of `x`. For `level = "groups"` it must be constant within each group, and for a trajectory model within each person.
`level`	`"individuals"` (default) relates the outcome to profiles, `"groups"` to group classes. For a trajectory model the outcome is related to its person classes, reported with `level = "persons"`; `"groups"` is refused there with `latents_bad_argument`.

method	"bch" (default) applies the Bolck-Croon-Hagenaars weights; "ml" fits the maximum-likelihood step-three model (see *Details*); "proportional" weights by the posterior probabilities without correction for classification error; "modal" uses the uncorrected modal assignment, for comparison.
ci_level	Confidence level for the intervals (default 0.95).
contrast	"none" (default) reports one outcome mean per class; "pairs" reports the difference between every pair of classes, with a test against zero.
adjust	Multiplicity correction across the pairwise differences, passed to stats::p.adjust(): "BH" (default), "holm", "hochberg", "hommel", "bonferroni", "BY" or "none". Ignored for contrast = "none".
vcov_type	"cluster" (default) sums the influence contributions within the fit's groups and treats the groups as the independent units, as required when the outcome is measured on observations nested in groups. "independent" treats every observation as an independent unit, which is appropriate only when observations can be treated as independent. The choice is recorded in the result's vcov_type attribute.
variance	For method = "ml": "varying" (default) estimates a variance of the outcome in each class, "equal" one variance shared by the classes. Given with another method it raises latents_bad_argument.

Details

On an lta() fit, level = "individuals" uses one row per occasion and relates the outcome to the profile at that occasion, with the default group-clustered variance accounting for the repeated occasions within a group. This is a marginal profile-outcome analysis, not a model of outcome effects on transitions or of a sequence-level outcome.

The cluster-robust variance is the sum of one outer product per independent unit, scaled by the small-sample (CR1) factor G / (G - 1) for G units (groups, or observations under vcov_type = "independent"). These contributions sum to zero at the estimate, so the variance has rank at most G - 1: it is zero with a single group and singular with as many groups as classes. Both cases raise latents_too_few_groups; vcov_type = "independent" gives the unclustered variance. For the same reason a class carried by a single unit (all its weight in one group, or under vcov_type = "independent" in one observation; Kish's effective number of carrying units below 1.5) has no standard error, whichever the method: its standard error, its interval and those of the contrasts involving it are NA, with a latents_small_classes warning.

With method = "ml" the assigned class is an indicator of the true class whose error probabilities are those of step one, and the outcome is normal within each true class. The log likelihood sum_i log sum_t pi_t D[t, W_i] phi(z_i; mu_t, sigma_t) is maximised over the class means and variances by EM followed by Newton steps, with the class sizes pi_t fixed at the average posterior probabilities and the classification errors D fixed, as in Latent GOLD's step-three model for a dependent variable. Multiple initialisations are used because the likelihood may have several local maxima. Equal-variance fits additionally use bounded BFGS optimisation from quantile-based initial values; accepted solutions must converge, have a sufficiently small score and yield a positive-definite information matrix. These searches do not guarantee the global maximum. Twenty random starts use a fixed internal seed, and the caller's .Random.seed is restored. R's hidden Box-Muller normal cache cannot be restored; see base::Random. Standard errors use a sandwich covariance estimator, clustered as for the other methods. As described above, a class supported by a single

independent unit has no standard error. Maximum-likelihood estimates depend on the assumed within-class outcome distribution. Skewness or an incorrect equal-variance constraint can bias estimated means; BCH is less sensitive to these distributional assumptions (Bakk and Vermunt, 2016).

Because uncertainty in the estimated classification-error probabilities is omitted, interval coverage can be below the nominal level, particularly when classification is uncertain.

Value

A base `data.frame`. For `contrast = "none"`, one row per class with the columns `level`, `method`, `class`, `estimate`, `standard_error`, `conf_low`, `conf_high` and `effective_n`; class means are not tested. For `method = "ml"`, a `variance` column (the outcome's estimated variance in the class) comes before `effective_n`, which is then the expected number of units in the class under the step-three model; the `variance` attribute records the choice. For `contrast = "pairs"`, one row per pair of classes with the columns `level`, `method`, `class`, `reference_class`, `estimate` (the mean of `class` minus the mean of `reference_class`), `standard_error`, `statistic`, `p_value`, `p_adjusted`, `conf_low` and `conf_high`, as in `r3step()`. A contrast with zero or missing standard error has `NA_real_` statistic and p-values.

The result's `vcov_type` attribute records the variance used; for `contrast = "pairs"` the `adjust` attribute records the correction applied to `p_adjusted`.

Interpretation

Correction for classification error depends on the fitted measurement model and assumes that the outcome and assigned class are independent conditional on latent class membership. The maximum-likelihood method additionally assumes a normal outcome distribution within each class. Reported uncertainty is conditional on the estimated classification-error probabilities and does not include uncertainty from estimating those probabilities or selecting the class solution. Outcome differences describe associations; they do not establish causal effects of class membership.

References

Vermunt, J. K. (2010). Latent class modeling with covariates: two improved three-step approaches. *Political Analysis*, 18, 450–469. 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, 272–311. Bakk, Z., & Vermunt, J. K. (2016). Robustness of stepwise latent class modeling with continuous distal outcomes. *Structural Equation Modeling*, 23, 20–31.

Examples

```
set.seed(11)
group <- rep(seq_len(40), each = 10)
truth <- 1L + as.integer(runif(400) > 0.5)
example_data <- data.frame(
  g = group,
  a = rnorm(400, ifelse(truth == 2L, 1.1, -1.1)),
  b = rnorm(400, ifelse(truth == 2L, 1.1, -1.1)),
  y = rnorm(400, ifelse(truth == 2L, 10, 0))
)
fit <- multilpa(example_data, c("a", "b"), "g", n_profiles = 2,
```

```
                n_group_classes = 1, n_starts = 4, seed = 1)
three_step(fit, example_data, "y")
three_step(fit, example_data, "y", contrast = "pairs")
```

`vcov.multilpa_covariates`

Extract model coefficients, covariance matrices and likelihood summaries

---

Description

These methods extract fitted parameters, measures of parameter uncertainty, and likelihood information for reporting and further statistical analysis.

`coef()` returns the model's parameter values. `vcov()` returns their estimated sampling variances and covariances, describing the uncertainty in individual estimates and their joint variation. `confint()` provides confidence intervals for selected parameters.

`logLik()` returns the fitted log likelihood together with the number of estimated parameters and the associated sample size. `nobs()` reports the number of independent units used in the model's group-based information criterion: individual observations for a single-level model and observed groups for a grouped model. The treatment of entirely missing units follows the fitted model's `drop_empty` setting.

Usage

```
## S3 method for class 'multilpa_covariates'
vcov(
  object,
  data = NULL,
  step = 1e-04,
  vcov_type = c("observed", "robust", "opg"),
  scale = c("natural", "unconstrained"),
  boundary = c("error", "fix"),
  ...
)

## S3 method for class 'multilpa'
coef(object, scale = c("natural", "unconstrained"), ...)

## S3 method for class 'multilpa'
vcov(object, data = NULL, scale = c("natural", "unconstrained"), ...)

## S3 method for class 'multilpa'
confint(object, parm, level = 0.95, data = NULL, ...)
```

Arguments

object	A fitted model: from multilpa() (class multilpa), a covariate model (multilpa_covariates), a latent transition model (multilpa_transitions), or pooled imputations (latents_pooled: coef(), vcov() and confint()).
data	Optional. The data frame the model was fitted to. When omitted it is rebuilt from what the fit stores (indicators, identifiers, occasions and designs), to floating-point precision.
step	Finite-difference step for the observed information (covariate models).
vcov_type	"observed", "robust" or "opg", as for parameter_inference() (covariate models).
scale	"natural", the default, or "unconstrained". Natural coefficients are expressed on the reported scale: means in indicator units, variances in squared indicator units, and probabilities on [0, 1]. Their covariance is obtained by the delta method. The unconstrained scale uses the coordinates in which the model is estimated: log variances (diagonal), log-Cholesky coordinates (full covariance), log volumes, shapes and orientations (a structure that constrains them across profiles), and baseline-category logits for mixing and response probabilities. In a covariate model, means and membership coefficients are the same on both scales. With method = "bootstrap" only the natural scale exists, and scale = "unconstrained" raises latents_unsupported_inference.
boundary	"error" or "fix", as for parameter_inference(): "fix" holds response probabilities on their bound, whose rows and columns are then zero (covariate models).
...	For vcov() and confint() of a multilpa fit, and confint() of a covariate model, passed to parameter_inference() (for example method = "bootstrap", vcov_type = "robust" or step); an argument parameter_inference() does not take raises latents_bad_argument. vcov() of a covariate model warns latents_unused_argument for an argument it does not take, and refuses method = "bootstrap" with latents_unsupported_inference; vcov() of a transition model warns latents_unused_argument for any. confint() of a transition model passes them to parameter_inference(). Ignored otherwise.
parm	Optional coefficient names or indices; defaults to every estimated coefficient. A coefficient held by fixed has no interval and raises latents_held_parameter; an unknown name or an out-of-range index raises latents_bad_argument.
level	Confidence level, strictly between 0 and 1 (default 0.95).

Details

These methods support profile and class models, models with membership covariates and homogeneous latent transition models. Pooled multiple-imputation results support coef(), vcov() and confint(). Methods for mixture regressions, growth mixtures, additive models, cross-level models and general transition models are described in get_results().

Categorical parameter names escape reserved characters within indicator and category components (%, :, ,, [and]) using percent codes. This keeps distinct parameters distinct when their original labels contain delimiters. Ordinary names are unchanged; use names(coef(object)) or numeric positions when selecting parameters for confint().

Value

`coef()`: a named numeric vector. `vcov()`: a named symmetric matrix over the same parameters. `confint()`: a matrix with one row per parameter and the lower and upper bounds as columns. `logLik()`: a `"logLik"` object with `df` and `nobs` attributes. `nobs()`: the number of independent groups counted by the fitted BIC. With `drop_empty = TRUE`, groups without any observed indicator are excluded; with `drop_empty = FALSE`, they are counted. Groups with zero sampling weight are excluded under either setting. For a single-level fit these groups are its individual observations.

See Also

`parameter_inference()` for the tidy table of estimates, standard errors and intervals.

Examples

```
fit <- multilpa(course_engagement, c("browse", "lectures", "forum_read"),
                "student", n_profiles = 2, n_group_classes = 2,
                n_starts = 2, seed = 1)
coef(fit)
logLik(fit)
nobs(fit)
```

Index

∗ **datasets**
course_engagement, 11
growth_schools, 47
growth_scores, 48
srl, 117
student_esm, 119
study_hours, 121
∗ **package**
latents-package, 3

as.data.frame(), *5*, *20*, *23, 24*, *26*, *45*, *78*, *103*, *118*

base::Random, *98*, *124*
bootstrap_lrt, *3*, 4
bootstrap_lrt(), *9*, *24*, *51*, *53*, *73, 74*, *89*, *91, 92*, *95*

candidate_fit *(*get_candidate*)*, 31
coef(), *78*
coef.multilpa *(*vcov.multilpa_covariates*)*, 126
compare_models, *3*, 8
compare_models(), *8*, *16*, *20*, *31*, *39*, *41*, *49*, *73*, *92*
confint(), *53*, *78*, *97*
confint.multilpa *(*vcov.multilpa_covariates*)*, 126
course_engagement, 11

descriptives, 13
descriptives(), *17*, *54*, *56*, *113, 114*
diagnostics, *3*, 15
diagnostics(), *15*, *39*, *41*, *45*, *50*, *113, 114*

enumerate_classes, *3*, 18
enumerate_classes(), *8–10*, *22–24*, *31*, *52*, *56*, *59*, *62*, *73*, *87*, *89*, *92*, *116*
enumerate_lca, *3*, 22
enumerate_lca(), *19*, *24*, *59*, *62*, *67*
enumerate_lpa, *3*, 23
enumerate_lpa(), *19*, *23*, *59*, *62*, *68*
enumerate_regressions, *3*, 24
enumerate_regressions(), *31*, *59*, *62*, *81*, *122*

fit_staged, *3*, 27
fit_staged(), *6*, *37*, *50*, *53*, *86*, *89*, *100, 101*, *115*, *119*

get_candidate, 31
get_candidate(), *9*, *18–21*, *23, 24*, *39*, *41*
get_group_tna, 32
get_group_tna(), *46*, *53, 54*
get_results, *3*, 33
get_results(), *17*, *19*, *48*, *54, 55*, *74*, *78, 79*, *89, 90*, *97*, *100*, *113, 114*, *119*, *127*
get_tna, 45
get_tna(), *33*, *53, 54*
growth_schools, 47
growth_scores, 48

latents *(*latents-package*)*, 3
latents-conditions, 49
latents-model-methods *(*vcov.multilpa_covariates*)*, 126
latents-package, 3
latents-plots, 57
latents-predict *(*predict.latents_mixture_regression*)*, 104
lca, *3*, 66
lca(), *9*, *14*, *22, 23*, *59*, *63*, *68*, *97*, *113*
logLik(), *78*
lpa, *3*, 67
lpa(), *9*, *14*, *24*, *59*, *63*, *67*, *97*, *113*
lta, *3*, 68
lta(), *5, 6*, *9*, *13, 14*, *19, 20*, *33*, *40*, *42*, *46*, *53*, *56*, *59*, *63, 64*, *91*, *97–99*, *110, 111*, *113*, *115*, *123, 124*

mixture_regression, *3*, 75

mixture_regression(), *9*, *25, 26*, *40*, *48, 49*, *55*, *59*, *99*, *110*, *115*, *122, 123*
multilca, *3*, 81
multilca(), *14*, *20*, *22*, *33*, *46*, *59*, *63*, *66, 67*, *95*, *97*, *106*, *113*
multilpa, *3*, 83
multilpa(), *5, 6*, *9*, *12–15*, *19–21*, *23*, *29, 30*, *33*, *46*, *53*, *56*, *59*, *63*, *66–68*, *70*, *72–74*, *82*, *97, 98*, *103*, *106–110*, *113*, *115*, *118–120*, *123*, *127*

nobs(), *78*

parameter_inference, *3*, 96
parameter_inference(), *34*, *37, 38*, *41*, *51–53*, *56*, *59*, *63*, *74*, *87*, *89*, *93*, *95*, *103*, *116*, *127, 128*
plot(), *5*, *20*, *23, 24*, *78, 79*
plot_diagnostics *(*latents-plots*)*, 57
plot_enumeration *(*latents-plots*)*, 57
plot_enumeration(), *26*, *54*
plot_prevalence *(*latents-plots*)*, 57
plot_profiles *(*latents-plots*)*, 57
plot_responses *(*latents-plots*)*, 57
plot_selection *(*latents-plots*)*, 57
plot_selection(), *54*
plot_sequences *(*latents-plots*)*, 57
plot_transitions *(*latents-plots*)*, 57
plot_tree *(*latents-plots*)*, 57
plot_tree(), *39*, *54*
plot_views *(*latents-plots*)*, 57
plot_views(), *114*
pool_imputations, *3*, 102
pool_imputations(), *40*, *53*, *59*
predict(), *78*, *91*, *93*
predict.latents_mixture_regression, 104
predict.multilpa *(*predict.latents_mixture_regression*)*, 104
prior_control, 107
prior_control(), *88*

r3step, *3*, 109
r3step(), *12, 13*, *37*, *51–53*, *85*, *89*, *91*, *125*
report, 113

sensitivity, *3*, 114
sensitivity(), *53*, *56*
set.seed(), *5*, *19*
simulate.latents_mixture_regression *(*predict.latents_mixture_regression*)*, 104
sort(), *29*, *70*, *86*
srl, 117
starting_values, 118
starting_values(), *30*, *53*, *89*
stats::anova(), *9*
stats::glm(), *72*
stats::p.adjust(), *41*, *97*, *103*, *110*, *124*
student_esm, 119
study_hours, 121
summary(), *5*, *17*, *20*, *23, 24*, *45*, *78*, *114*

three_step, *3*, 122
three_step(), *36, 37*, *51–53*, *55*, *85*, *89*, *91*, *110*, *112*
tna::tna(), *33*, *46*
tryCatch(), *49*

utils::write.csv(), *34*

vcov(), *53*, *78*, *97*
vcov.multilpa *(*vcov.multilpa_covariates*)*, 126
vcov.multilpa_covariates, 126

withCallingHandlers(), *49*