

Package: latents (via r-universe)

October 16, 2026

Title Multilevel Latent Profile, Class and Transition Models

Version 0.9.21

Description Provides functions for fitting multilevel latent profile analysis, latent class, and latent transition models for continuous, categorical, and mixed-type variables. Models are estimated by expectation-maximization, handle missing data, and provide standard and robust standard errors. Model selection is supported by information criteria, entropy, classification diagnostics, and a parametric bootstrap likelihood ratio test. Covariates can be related to latent class membership using three-step methods with correction for classification error. Latent transition models estimate transitions between latent profiles across ordered occasions and are compatible with the 'tna' package for network and sequence analysis. The multilevel mixture formulation follows Vermunt (2003)
<[doi:10.1111/j.0081-1750.2003.t01-1-00131.x](https://doi.org/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, graphics, grDevices, utils

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

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 <https://cran.r-universe.dev>

Date/Publication 2026-10-16 09:20:18 UTC

RemoteUrl <https://github.com/cran/latents>

RemoteRef HEAD

RemoteSha 1a2dba1117e4c9466b58af664069066cc6db2f79

Contents

as.data.frame.mutilpa	3
bootstrap_lrt	5
candidate_fit	7
compare_models	8
course_engagement	9
descriptives	11
diagnostics	14
enumerate_classes	15
enumerate_lca	18
enumerate_lpa	19
enumerate_regressions	20
fit_staged	22
get_group_tna	25
get_results	26
get_results.latents_family_enumeration	33
get_results.latents_growth_mixture	34
get_results.latents_mixture_regression	36
get_results.latents_pooled	39
get_results.latents_transition_enumeration	40
get_results.mutilpa_additive	41
get_results.mutilpa_cross_level	43
get_results.mutilpa_lta	45
get_tna	47
growth_schools	48
growth_scores	50
latents-conditions	51
lca	56
lpa	57
lta	59
mixture_regression	63
mutilca	68
mutilpa	70
parameter_inference.mutilpa_additive	79
parameter_inference.mutilpa_transitions	85

plot.latents_comparison	87
plot.latents_family_enumeration	88
plot.latents_growth_mixture	89
plot.latents_mixture_regression	90
plot.latents_pooled	91
plot.latents_transition_enumeration	92
plot.mutilpa	93
plot.mutilpa_additive	96
plot.mutilpa_bootstrap_lrt	97
plot.mutilpa_covariates	98
plot.mutilpa_cross_level	99
plot.mutilpa_enumeration	100
plot.mutilpa_lta	102
plot.mutilpa_transitions	102
plot_views	104
pool_imputations	104
predict.latents_growth_mixture	107
predict.latents_mixture_regression	107
predict.mutilpa	108
print.mutilpa_enumeration	110
prior_control	112
r3step	113
report	115
sensitivity	117
simulate.latents_growth_mixture	118
simulate.latents_mixture_regression	119
srl	120
starting_values	121
student_esm	122
study_hours	124
summary.mutilpa_additive	125
three_step	127
vcov.mutilpa_covariates	129

Index**132**

as.data.frame.mutilpa

Convert latents results to a data frame

Description

as.data.frame() returns the primary table of a latents result: the measurement parameters of a fitted model, the candidates of an enumeration, the test of a bootstrap, and so on. It is the same table as get_results(x); [get_results\(\)](#) with what = returns the others.

Usage

```
## S3 method for class 'mutilpa'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'mutilpa_enumeration'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'mutilpa_start'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'mutilpa_covariates'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary_mutilpa_additive'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary_mutilpa_covariates'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary_mutilpa_enumeration'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'mutilpa_bootstrap_lrt'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary_mutilpa_bootstrap_lrt'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'latents_table'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary_mutilpa'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'mutilpa_transitions'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'summary_mutilpa_transitions'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	An object returned by a latents function, or the summary() of one.
row.names	Passed to data.frame() ; NULL gives default row names.
optional	Ignored; present for compatibility with as.data.frame() .
...	Must be empty. An argument here raises latents_bad_argument naming it, rather than being dropped.

Value

A base `data.frame`, one row per parameter, candidate or test depending on the object.

See Also

[get_results\(\)](#), [latents-print](#), [latents-summary](#).

Examples

```
fit <- multilpa(course_engagement, c("browse", "lectures", "forum_read"),
               "student", n_profiles = 2, n_group_classes = 2,
               n_starts = 2, seed = 1)
head(as.data.frame(fit))
```

bootstrap_lrt

*Parametric bootstrap likelihood-ratio comparison***Description**

Simulates indicators under the null model while preserving observed group sizes, refits both models, and compares their likelihood differences. Models must differ by exactly one individual profile or one group class, with the other count, covariance structure and centering mode fixed. Grand-mean centering is repeated in every simulated refit. Person-centred fits are refused because this model does not specify a generative distribution for the group baselines removed by that transformation. This is a native parametric bootstrap, not an implementation of Mplus TECH14. It does not use a chi-square reference distribution. Any failed/nonconverged or reversed replicate makes the p-value NA, avoiding silent deletion of difficult fits.

Usage

```
bootstrap_lrt(
  null_model,
  alternative_model,
  data = NULL,
  iter = 199L,
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  seed = NULL
)
```

Arguments

`null_model` Smaller, converged [multilpa\(\)](#) model, with or without membership covariates.
`alternative_model` Larger model fitted to exactly the same data.

<code>data</code>	Optional. The data both models were fitted to, used to verify both fitted likelihoods; when omitted it is rebuilt from what the null model stores.
<code>iter</code>	Number of simulated datasets (at least two; use many for inference).
<code>n_starts</code>	Number of starts for each simulated fit.
<code>max_iter</code>	Maximum EM iterations for each simulated fit.
<code>tol</code>	Relative likelihood convergence tolerance.
<code>seed</code>	Optional seed: any whole number <code>set.seed()</code> accepts. The caller's random-number state is restored on exit.

Details

Fits made with `missing = "fiml"` are supported when both models were: every replicate is given the observed data's missing cells before it is refitted, preserving the pattern of available measurements. Calibration requires missingness independent of the indicators and latent classes, conditional on any fixed covariates. Carrying a fixed mask does not reproduce a general MAR mechanism that depends on observed indicators, or an MNAR mechanism. FIML estimation under MAR does not by itself validate this bootstrap under MAR; the retained size study covers MCAR only.

Membership-covariate fits (`multilpa(profile_covariates = , group_covariates = )`) are supported: the covariates are held at their observed values, group classes are drawn from the fitted group logits and profiles from each row's profile logits, as the model conditions on the covariates. Both models must use the same profile covariates, group covariates and slope specification, except that a one-group-class null, which cannot carry group covariates or slopes by group class, may be compared with an alternative that adds them. When data is omitted it is rebuilt from the alternative, whose columns include the null's. A covariate replicate counts as valid when its likelihood has converged even if a membership logit is still drifting, as it does for an empty or separated class in an over-fitted alternative: the statistic reads only the maximized likelihood, which such a fit has reached. The replicate table's `logits_settled` column records whether both refits' logits had also converged. The original models must be fully converged.

Models fitted with `fixed`, including those from `fit_staged()`, are supported: every replicate is refitted with the same blocks held at the same values, so the simulated statistics compare the same two constrained models as the observed statistic does. The p-value is then conditional on that measurement solution, which is treated as known and whose own uncertainty is not propagated. A constrained pair whose nesting cannot be established is refused rather than given a p-value.

Value

An object of class `multilpa_bootstrap_lrt`, carrying the observed statistic, the finite-simulation corrected p-value, its Monte Carlo standard error, the measurement blocks held fixed in every fit, and one record per replicate. Read it with the verbs that describe it: `as.data.frame()` gives the single-row test result, `get_results(what = "replicates")` one row per simulated replicate, `summary()` the test beside the replicate diagnostics, and `plot()` the simulated null distribution with the observed statistic marked. Inspect failed starts, boundary flags and likelihood replication before interpreting results.

Conditions

`latents_bad_nesting` when only one of the two models holds measurement blocks fixed, when they hold different blocks, when they hold the same blocks at different values, when they hold a

measurement fixed while differing in the number of profiles (the held blocks then have different shapes, so the null is not a restriction of the alternative), or when the alternative does not estimate more free parameters than the null. `latents_failed_replicates` is warned when some replicate fails validation, and the p-value is NA. `latents_unsupported_bootstrap` refuses person-centred fits, for which this simulator has no group-baseline distribution. `latents_unsupported_prior` refuses fits estimated with a prior: their likelihoods are evaluated at posterior modes, not ML estimates. `latents_incomparable_models` when the two models differ in their observations, grouping, covariance structure, centering or missing-data handling; `latents_bad_data` when the data have missing indicators and the models were not fitted with `missing = "fiml"`; `latents_bad_nesting` also when two covariate models use different covariates or slope specifications (other than a one-group-class null).

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
```

candidate_fit

Take one fitted model out of an enumeration grid

Description

An enumeration keeps every candidate it fitted. This returns one of them, so that a caller who wants to plot, summarise or test a particular candidate names it by its class counts instead of indexing the grid by position.

Usage

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

Arguments

<code>x</code>	An <code>multilpa_enumeration</code> result from <code>enumerate_classes()</code> .
<code>n_profiles</code>	Number of individual profiles identifying the candidate.
<code>n_group_classes</code>	Number of group classes identifying the candidate.

model The covariance model identifying the candidate, needed when `enumerate_classes()` was given several models and the class counts alone name several candidates. Naming counts that match more than one candidate raises `latents_unknown_candidate` listing the models it could have meant.

Value

The fitted model for that cell of the grid: an object of class `multilpa`, exactly as `multilpa()` returned it, with every verb of this package available on it.

Conditions

`latents_unknown_candidate` when the requested class counts are not in the grid, and `latents_failed_candidate` when they are in the grid but that fit raised an error, so no model exists to return. Neither situation returns `NULL`, because a `NULL` would flow silently into whatever the caller did next.

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)
candidate_fit(candidates, n_profiles = 2, n_group_classes = 1, model = "VVI")
```

compare_models

Compare fitted models on the same data

Description

One row per model with its log likelihood, number of parameters, information criteria, classification quality and convergence, so alternative specifications (for example trajectory classes alone against a growth mixture model) are compared without assembling tables by hand. `delta_bic` is each model's BIC above the smallest, and `bic_weight` the Schwarz weight, $\exp(-\text{delta}/2)$ normalized over the models compared (Wagenmakers and Farrell 2004): the approximate posterior probability of each model if one of them is true, with equal prior probabilities.

Usage

```
compare_models(...)
```

Arguments

... Two or more fits from `mixture_regression()` (with or without random). Name them to label the rows; unnamed fits are labelled by their expression.

Details

Information criteria are comparable only on the same data: every model must be fitted to the same outcome, persons (or groups) and observations, or the comparison is refused.

Value

A base data.frame of class `latents_comparison` with one row per model: `model`, `n_classes`, `random`, `n_parameters`, `log_likelihood`, `aic`, `bic`, `sabic`, `icl`, `entropy`, `smallest_share`, `delta_bic`, `bic_weight`, `converged`. Raises `latents_bad_argument` for fits it cannot compare and `latents_incomparable_models` for fits to different data.

References

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

Examples

```
# Fixed trajectories against growth curves with a random intercept,
# for the first 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", seed = 1)
compare_models(trajectories = trajectories, growth = growth)
```

course_engagement

Student engagement across a sequence of courses

Description

A simulated dataset with the structure this package exists for, and all of it in one place: enrolments nested in students, ordered within each student, with a covariate that predicts which pattern an enrolment follows.

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. Which course the enrolment was in.

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

browse Numeric, course page views on the $\log_{10}(\text{events})$ scale, standardized within course.

lectures Numeric, lecture videos viewed on the $\log_{10}(\text{events})$ scale, standardized within course.

- forum_read** Numeric, forum posts read on the $\log_{1p}(\text{events})$ scale, standardized within course.
- forum_post** Numeric, forum posts written on the $\log_{1p}(\text{events})$ scale, standardized within course.
- attendance** Numeric, days the student was active in the course as on the $\log_{1p}(\text{events})$ scale, standardized within course. It is **generated from the four click measures** and then blurred, because a student is recorded present on a day precisely because they clicked something, and a day is a coarse unit – one tick however much happened inside it. The indicators are therefore locally dependent on purpose: attendance correlates 0.51 to 0.67 with the click measures overall and 0.16 to 0.26 within a single profile, which is more association than a shared profile alone would produce. See the note on local dependence below.
- previous_grade** Numeric, the grade the student earned in the course *before* this one, standardised across the cohort. At a student's first course an entry grade stands in its place. It is a predictor of engagement here, not an outcome of it: pass it as `profile_covariates` to `multilpa(profile_covariates = )`, or as the predictor to `r3step()`.
- engagement** Factor, "disengaged" or "engaged", the pattern the row was generated from. This is the truth behind the data, not something a study would observe; it is shipped so a fitted model can be checked against what produced it. It is deliberately not called `profile`: `get_results(fit, "assignments")` adds a column of that name, and a dataset already carrying it would make the verb refuse to join its own output.
- student_type** Factor, "committed" or "wavering", the kind of student the row's student was generated as. The truth behind the group classes, shipped for the same reason.

Details

Each row is one student's enrolment in one course. A student takes up to fifteen courses, and sequence orders them, so the same rows support a cross-sectional two-level model (`id = "student"`), a latent transition model (`time = "sequence"`), and a membership-covariate model (`profile_covariates = "previous_grade"`).

Two kinds of student were generated. Both contain both patterns of engagement; they differ in the *mix*, which is the two-level quantity this package estimates. A committed student is engaged in about four enrolments in five, a wavering student in about one in four. Pooled across the sample the engaged share describes neither kind, which is the point: recovering the split is what a two-level model does and a pooled one cannot.

Local independence does not hold here, on purpose

A latent profile model with a diagonal residual covariance assumes the indicators are independent given the profile. `attendance` is built from the click measures, so for these data that assumption is false, and the package finds it: `get_results(fit, "residuals")` reports a largest residual of 0.21 $p = 1.7e-15$, and `covariance_model = "full"` improves the log likelihood from -7618.5 to -7494.7 for twenty more parameters, which every information criterion prefers.

This is deliberate. A bundled dataset that satisfies local independence exactly can only ever show those verbs agreeing that nothing is wrong; here the diagnostic has a real violation to find, its cause is known, and the fix can be seen to work. One consequence is worth knowing before it surprises you: under a diagonal model `enumerate_classes()` keeps preferring more classes, because extra classes absorb residual association the model is not allowed to express directly.

The five activity indicators are **standardized within course**: each value is that enrolment's distance from its own course's mean, in that course's standard deviations. Every indicator therefore has mean

zero and standard deviation one within every course, and a browse of 1.2 reads as "1.2 standard deviations more active than the average enrolment in that course". This is the scale a learning-analytics reader meets, because activity volume is not comparable across courses: a course with weekly quizzes generates more clicks than a seminar whatever its students are like, so an analyst compares a student with their coursemates rather than with the whole cohort.

In this simulation the courses do not in fact differ in activity volume, so the standardization sets the scale rather than removing a course effect. It is stated here rather than left implicit because a reader who assumed otherwise would misread what the course column does.

The indicators are simulated directly on the `log1p(events)` scale, the scale of a logged activity export; no raw counts are generated. They are standardized on that scale, not as counts. Standardizing is affine within a course, so it leaves the within-course skew exactly as it found it, and raw learning-analytics counts are strongly right-skewed: on an unlogged draft of these data the skew alone looked like an extra class, exactly as a floor or ceiling effect does. The log removes the skew; the standardization puts the courses on a common footing afterwards.

Source

Simulated by `data-raw/course-engagement.R`, seed 2026. The simulation is calibrated to the shape of a real learning-analytics export – its scales, its spreads, its correlations and how persistent engagement is across a student's courses – but only those aggregate constants were used. No row, identifier or value of any real student is present, and the source is 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 `multilpa(profile_covariates = )` for `previous_grade`.

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")
# The generating truth ships alongside, so recovery can be checked directly.
get_results(fit, what = "assignments", data = course_engagement,
  truth = "engagement")
```

Description

One row per variable, with the summaries you would look at before fitting and the one that decides whether a two-level model has anything to find: the intraclass correlation, the share of each variable's variance that lies *between* groups rather than within them.

Usage

```
descriptives(x, ...)

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

## S3 method for class 'multilpa'
descriptives(x, by = NULL, ...)

## S3 method for class 'multilpa_transitions'
descriptives(x, by = NULL, ...)

## S3 method for class 'multilpa_covariates'
descriptives(x, by = NULL, ...)
```

Arguments

<code>x</code>	A data frame, or a fitted model of this package. A fit carries the columns it was built from, so <code>vars</code> and <code>id</code> are read from it.
<code>...</code>	Passed between methods.
<code>vars</code>	Character vector of column names to describe. Repeated names are described once. For a data frame, defaults to every numeric column that is not <code>id</code> .
<code>id</code>	Optional. The column identifying the groups rows are nested in. Supplying it adds <code>n_groups</code> and <code>icc</code> ; omitting it leaves them out.
<code>by</code>	Optional. "profile" or "group_class" on a fitted model, or a column name on a data frame: describe each variable once per level of it, rather than once overall. Rows whose stratifier is <code>NA</code> are neither dropped nor folded into another level: they form their own stratum, labelled <code>NA</code> in the <code>by</code> column and reported last, so that the strata account for every input row exactly once. The stratifier is never compared with <code>==</code> , so a missing value cannot select rows it does not belong to.

Details

An ICC near zero suggests little between-group mean variation in that variable. It does not rule out group classes that differ in other variables, profile prevalence, or other features of their distributions.

Value

A base `data.frame`, one row per variable, or one row 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, and `NA_real_` for a variable that is not numeric.

`n_distinct` integer: distinct observed values, which is the useful summary where a mean is not.

`n_groups,icc` present only when `id` is known. `icc` is the one-way random-effects estimate, and may be slightly negative when the between-group mean square falls below the within-group one; that is the estimator reporting no group structure, not an error. Rows with a missing group ID do not contribute to either statistic.

A by column comes first, named after what it splits on and holding that level's own value, NA for the stratum of rows whose stratifier is missing. For every variable, `n + n_missing` summed over the strata equals the number of input rows; that invariant is asserted before the table is returned. Describing by a class the model assigned keeps that assignment out of the described frame, so an indicator of the caller's own that is named `profile` or `group_class` is summarized as itself, not replaced by the assignment. A by whose name is one of the summary's own columns cannot be told apart from them and raises `latents_bad_data`.

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()` to fit, `diagnostics()` for the after-the-fit counterpart.

Examples

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

# A missing stratifier is a stratum of its own, not a silent loss and not
# missingness invented in a variable that has none.
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

Every classification diagnostic, in one call

Description

`summary()` reports what a model estimated. This reports whether to believe it: how sharply the posterior separates the classes, how big each class actually is, how confidently each unit was assigned, and whether the within-profile independence the model assumes survives contact with the data.

Usage

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

```
## S3 method for class 'multilpa_diagnostics'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

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

```
## S3 method for class 'multilpa_diagnostics'
plot(x, ...)
```

Arguments

<code>x</code>	A fitted model of this package.
<code>data</code>	Optional. The data the model was fitted to. A fit carries the columns it was built from, so this is only needed to override them.
<code>plots</code>	TRUE, the default, also draws the four classification plots (effective profile sizes, posterior probabilities of the assigned profiles, case entropy, and the average posterior probability matrix), as printing <code>plot()</code> on the result does. Without the <code>ggplot2</code> package a message says so and the tables are still returned. FALSE returns the tables only.
<code>by</code>	"profile", the default, assesses each profile's bivariate residuals separately; "overall" pools them. It is the one argument of a gathered table this function forwards, because it is the one that changes what a gathered table means rather than which fit it is taken from. The level the classification tables use is not an argument here: it follows from whether the fit has discrete group classes.
<code>...</code>	For <code>diagnostics()</code> , nothing further is accepted. An argument this function cannot forward raises an error of class <code>latents_bad_argument</code> naming it, rather than being dropped on the way to a table that then means something other than what was asked for. For <code>plot()</code> , nothing further is accepted either; style the returned plots with <code>ggplot2</code> .
<code>row.names, optional</code>	Passed to the base generic; <code>row.names</code> is applied to the returned table.

Details

It gathers the four classification tables of `get_results()` — "entropy", "classification", "average_posteriors" and "residuals" — and prints one line of reading per diagnostic rather than four differently shaped tables. The tables themselves come back from `get_results()` on the result, or on the fit.

Value

An object of class `multilpa_diagnostics`. Read its tables with `get_results(result, what = )`, which offers "entropy", "classification", "average_posteriors", "residuals" and "all", and never with `$`. A model family that has no bivariate residuals leaves that table out of "all", and asking for it by name raises `latents_no_group_classes`.

`print()` returns the object invisibly, having printed one line per diagnostic: relative entropy, smallest class, lowest average posterior and largest residual, at each level the fit has. `plot()` returns every classification plot as a `latents_plots` list of `ggplot` objects – the effective profile sizes, the posterior probability of each case's assigned profile, each case's relative entropy, and the average posterior probability matrix – that draws all four when printed. `as.data.frame()` returns the entropy table, the primary one.

See Also

`get_results()` for these tables and every other one, `descriptives()` for the before-the-fit counterpart, and `summary()` for what the model estimated rather than whether to trust it.

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)
quality
get_results(quality, what = "classification")
get_results(diagnostics(fit, by = "overall"), what = "residuals")
```

enumerate_classes

Enumerate numbers of individual profiles and group classes

Description

Fits every requested combination and retains errors and warnings alongside successful fits. Information criteria are descriptive: neither the smallest BIC nor high entropy guarantees the correct number of classes. Unconverged, boundary, and unreplicated fits are reported, not silently selected.

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,
  ...
)

```

Arguments

<code>data</code>	Data frame.
<code>vars</code>	Continuous indicator names.
<code>id</code>	The column that identifies the groups the observations are nested in (students in schools, reports in students). <code>NULL</code> enumerates single-level models, with a message; enumerate_lpa() and enumerate_lca() fit those by name.
<code>n_profiles</code>	Positive integer profile counts to try.
<code>n_group_classes</code>	Positive integer group-class counts to try. With <code>id = NULL</code> it is 1 and may be left out.
<code>model</code>	The covariance structures to cross with the class counts. "basic", the default, fits the four structures that combine variances equal or varying across profiles with covariances absent or present: "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). They are the choices that matter most in practice, since omitting covariances between correlated indicators makes the grid favour additional profiles. "all" fits all 14 structures of Celeux and Govaert (1995). Otherwise name the structures by their three-letter codes, as multilpa() 's <code>model</code> 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. <code>NULL</code> crosses no structures and fits whatever the other arguments ask for. With one continuous indicator the structures reduce to equal or varying variance, and codes that coincide are fitted once; with only categorical indicators there is no structure to cross. A structure set through <code>variance_model</code> , <code>covariance_model</code> , <code>volume</code> , <code>shape</code> or <code>orientation</code> replaces the default.
<code>seed</code>	Optional reproducible seed for each fit.
<code>family</code>	"profiles" (the default) enumerates the profile model over <code>n_profiles</code> , <code>n_group_classes</code> and <code>model</code> . One or more of the group-class families "additive", "dispersion" and "additive_dispersion" enumerates those instead, over <code>family</code> , <code>n_group_classes</code>

and `between_variance`; `n_profiles` and `model` are then refused, and the result is a `latents_family_enumeration` read with `get_results.latents_family_enumeration()` and `candidate_fit()`.

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".

time

The occasion column: enumerate latent transition models (`lta()`) over `n_profiles` (default 2:4) and `n_group_classes` (default 1), holding every other `lta()` argument given in `...` fixed. `model`, when given, crosses those covariance structures. The result is a `latents_transition_enumeration`, read with `get_results.latents_transition` and `candidate_fit()`.

...

Further arguments to `multilpa()` (or to `lta()` with time); for group-class families only `n_starts`, `max_iter`, `tol` and `min_variance`.

Value

An object of class `multilpa_enumeration`. Read it with the verbs that describe it rather than by reaching into it: `as.data.frame()` gives one row per candidate model with every criterion and diagnostic, `summary()` gives one row per information criterion naming the candidate that minimises it, `plot()` draws one criterion across the grid, and `candidate_fit()` returns the fitted model for one cell of the grid.

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 every indicator is categorical.

References

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

See Also

`candidate_fit()` to take one fitted model out of the grid, `summary.muiltlpa_enumeration()` 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)
```

[illegible]

```

n_profiles = 1:3, model = c("VVV", "EEE"),
n_starts = 2, seed = 1)

summary(single)

```

enumerate_lca

Compare latent class models

Description

Fits single-level latent class models with each number of classes in `n_classes` and returns them in one table for comparison by information criteria, entropy and the diagnostics of each fit. Every indicator is categorical unless named in ordinal or count, as in `lca()`. It is `enumerate_classes()` with `id = NULL`, and it returns the same object; the number of classes is its `n_profiles` column.

Usage

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

Arguments

<code>data</code>	A data frame with one row per observation.
<code>vars</code>	Names of the categorical indicator columns.
<code>n_classes</code>	Positive whole numbers of classes to compare.
<code>...</code>	Further arguments for <code>multilca()</code> , such as <code>missing</code> , <code>n_starts</code> , <code>seed</code> and <code>min_probability</code> , except <code>id</code> , <code>n_group_classes</code> and <code>categorical</code> .

Details

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 each criterion prefers, `plot()` the criteria against the number of classes, and `candidate_fit()` one fitted candidate, named by its `n_profiles`.

Conditions

`latents_bad_argument` when `id`, `n_group_classes`, `categorical` or `model` is passed.

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	<i>Compare latent profile models</i>
---------------	--------------------------------------

Description

Fits single-level latent profile models with each number of profiles in `n_profiles` under each covariance structure in `model`, and returns them in one table for comparison by information criteria, entropy and the diagnostics of each fit. It is [enumerate_classes\(\)](#) with `id = NULL`, and it returns the same object.

Usage

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

Arguments

<code>data</code>	A data frame with one row per observation.
<code>vars</code>	Names of the indicator columns.
<code>n_profiles</code>	Positive whole numbers of profiles to compare.
<code>model</code>	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() .
<code>...</code>	Further arguments for multilpa() , such as <code>missing</code> , <code>n_starts</code> , <code>seed</code> and <code>categorical</code> , except <code>id</code> and <code>n_group_classes</code> .

Details

For observations nested in groups, use [enumerate_classes\(\)](#) with the grouping column as `id`, which also crosses 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 each criterion prefers, [plot\(\)](#) the criteria against the number of profiles, and [candidate_fit\(\)](#) 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 *Compare mixture regressions with different numbers of classes*

Description

Fits [mixture_regression\(\)](#) for every requested number of classes (and of group classes, for the two-level model) and returns their fit statistics in one table. Optionally adds a parametric bootstrap likelihood-ratio test of $k - 1$ against k classes (McLachlan 1987; Nylund, Asparouhov and Muthen 2007): outcomes are simulated from the $k - 1$ class fit, both models are refitted to every replicate, and the p-value is the share of replicate statistics at least as large as the observed one, $(1 + b) / (1 + B)$.

Usage

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

## S3 method for class 'latents_regression_enumeration'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'latents_regression_enumeration'
```

```
get_results(
  x,
  what = c("fit", "model"),
  n_classes = NULL,
  n_group_classes = 1L,
  ...
)

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

Arguments

formula	A model formula outcome ~ predictors. Factors, interactions and offset() are supported.
data	A data frame.
n_classes	Integer vector of class counts to fit.
n_group_classes	Integer vector of group-class counts (two-level model; needs id). Crossed with n_classes.
bootstrap	Number of bootstrap replicates for the likelihood-ratio test; 0 skips it.
bootstrap_starts	Random starts for each bootstrap refit.
seed	NULL or an integer seed. The caller's random state is restored afterwards.
...	Further arguments to mixture_regression() , such as family, id, class_level, common, membership or variance.
x	A latents_regression_enumeration object.
row.names, optional	Unused; part of the generic.
what	"fit" for the comparison table, "model" for one fitted model.
digits	Significant digits.

Value

An object of class `latents_regression_enumeration`. Its table (from [as.data.frame\(\)](#) or `get_results(x, "fit")`) has one row per model: `n_classes`, `n_group_classes`, `log_likelihood`, `n_parameters`, `aic`, `bic`, `bic_rows`, `sabic`, `icl`, `entropy`, `smallest_share`, `converged`, `n_best_replicated`, `best_bic` (the minimum-BIC row), and, with `bootstrap > 0`, `blrt_statistic`, `blrt_p_value` and `blrt_replicates` (the successful replicates) and `blrt_flagged` (the failed, unconverged, separated or reversed-likelihood replicates). The p-value is NA if any requested replicate fails validation. `get_results(x, "model", n_classes = , n_group_classes = )` returns one fitted model.

Conditions

`latents_bad_argument` for an empty or invalid grid. A model that cannot be fitted at some count (for example every start degenerates) gets a row of NA with its condition message in note, rather than stopping the comparison.

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

with_test <- enumerate_regressions(score ~ hours, data = study_hours,
                                   n_classes = 1:2, n_starts = 3, seed = 1,
                                   bootstrap = 19)

as.data.frame(with_test)
```

fit_staged

Fit a two-level model in stages, holding the measurement solution

Description

Estimates the measurement model once, on its own, and then estimates the group-class structure with that measurement held fixed. This is the staged workflow used when the measurement solution is meant to be decided before the grouping question is asked, so that adding group classes cannot move the profiles they are meant to describe.

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 = 1e-06,
  seed = NULL,
  missing = c("error", "fiml"),
  covariance_model = c("diagonal", "full"),
  categorical = character(),
```

```

    min_probability = 1e-10,
    time = NULL
  )

```

Arguments

<code>data</code>	A data frame containing indicators and a group identifier.
<code>vars</code>	Unique character vector of continuous indicator column names.
<code>id</code>	Name of the observed group identifier column. Character, factor, or numeric identifiers are supported; missing identifiers are not. Required here: this model has a second level by construction, so it has no single-level form and does not take <code>multilpa()</code> 's <code>id = NULL</code> .
<code>n_profiles</code>	Positive integer number of individual profiles.
<code>n_group_classes</code>	Number of latent group classes to estimate in the second stage. Must be at least two; one would leave nothing to estimate.
<code>measurement</code>	Optional fitted <code>multilpa</code> model to use as the first stage, so that a measurement solution already chosen and inspected is carried forward rather than refitted. Its profiles, indicators and measurement options must match the ones requested here, and it must have one group class: a multi-class fit already estimates group structure and is not a measurement-only first stage. When <code>NULL</code> , the first stage is fitted here with one group class.
<code>variance_model</code>	Either "varying" (profile-specific variances or covariance matrices) or "equal" (shared across profiles).
<code>n_starts</code>	Number of EM starts, used for both stages.
<code>max_iter</code>	Nonnegative integer maximum number of EM updates per start. <code>max_iter = 0</code> performs no update. With <code>start</code> , it returns the model evaluated at exactly those values, whatever <code>n_starts</code> is, so that <code>logLik()</code> scores a parameter set supplied from elsewhere rather than the best of some random initializations that were never asked for. Without <code>start</code> there is nothing to evaluate at, so the random initializations are scored and the highest is returned.
<code>tol</code>	Positive relative log-likelihood tolerance. Convergence requires absolute change no greater than <code>tol * (1 + abs(previous log likelihood))</code> .
<code>min_variance</code>	Positive lower bound on each variance, or each covariance eigenvalue for full covariance, in squared input units. This defines a constrained maximum-likelihood problem. Bound-active estimates are explicitly reported and generate a warning.
<code>seed</code>	Optional random seed: any whole number set. <code>seed()</code> accepts, negative ones included. With a supplied seed, the caller's random-number state is restored on exit.
<code>missing</code>	"error" rejects missing indicators; "fiml" maximizes the observed-data likelihood under an ignorable missingness mechanism (MAR). Missing indicators are integrated out, not filled in for likelihood fitting. This holds with membership covariates too; the covariates themselves must be complete (<code>latents_bad_data</code>).
<code>covariance_model</code>	"diagonal" 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, which is the latent class measurement model. Binary, ordinal and unordered indicators are all handled by the same unrestricted 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, defining a constrained maximum-likelihood problem in the same way <code>min_variance</code> 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. The model does not use it; it is stored so that <code>get_results(x, "sequences")</code> , <code>get_results(x, "sequence_summary")</code> and <code>plot(what = "sequences")</code> can read the assignments back in order. Values must be complete and unique within each group.

Details

The second stage treats the measurement solution as known, so its Wald standard errors (`parameter_inference()`'s default), information criteria and likelihood-ratio comparisons are conditional on that solution and are narrower than they would be if the measurement had been estimated jointly. `parameter_inference(fit, method = "bootstrap")` removes that conditioning for the standard errors and intervals: every re-sample of the groups is staged again, measurement first, so the replicates carry the first stage's sampling variability into the second stage, and the table also reports the measurement parameters with their own errors. A measurement supplied by the caller is re-estimated on each resample with the same options, since the resample is new data. The conditioning of the criteria and likelihood comparisons is a property of staging itself, and it is the reason both parameter counts are reported: compare staged fits with one another using `n_parameters`, and compare a staged fit with a jointly estimated one using `n_parameters_with_measurement`, remembering that the staged likelihood is not the joint maximum and the comparison is descriptive.

Value

A `multilpa` object for the second stage, so that every accessor, diagnostic and method works on it unchanged. Its measurement parameters are exactly the first stage's. It additionally carries `fixed`, naming the held blocks; `n_parameters`, counting only the parameters this stage estimated; and `n_parameters_with_measurement`, which adds the held measurement back. Use `get_results()` with `what = "stages"` for a tidy two-row summary of both stages.

See Also

`multilpa()` with `fixed` for finer control over which blocks are held, and `starting_values()` with `what = "measurement"` for the values a stage hands on.

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_group_tna

A transition network for each latent group class

Description

Builds one `tna::tna()` model per latent group class of a fitted transition model, collected into the `group_tna` object `tna`'s grouped verbs expect. A class row with no expected outgoing moves keeps the fit's unestimated transition probabilities and raises `latents_empty_transition_row`.

Usage

```

get_group_tna(x, ...)

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

## S3 method for class 'multilpa_lta'
get_group_tna(x, ...)

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

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

```

Arguments

<code>x</code>	A fitted model from <code>lta()</code> , or a <code>multilpa()</code> / <code>multilca()</code> fit made with <code>time =</code> , whose modal profile sequences are then used.
<code>...</code>	Passed to <code>tna::tna()</code> for each class.
<code>label</code>	What the classes are called in <code>tna</code> 's output.

Value

An object of class `group_tna`, one `tna` model per latent class: `centralities()` returns one tidy table with a group column, `compare()` contrasts two classes, and `plot()` draws them together.

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

Every table a fitted model holds, from one verb

Description

One verb returns every tidy table this package computes, named by what. The tables were once eleven separate verbs and seven `as.data.frame()` methods with their own what catalogues; a reader had to know which verb owned which table before they could ask for it. Here the object decides what it can offer and what names it, so the same call shape reads a measurement model, a classification diagnostic and a transition matrix.

Usage

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

## Default S3 method:
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa'
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa_covariates'
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa_transitions'
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa_enumeration'
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa_bootstrap_lrt'
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa_start'
get_results(x, what = NULL, ...)

## S3 method for class 'multilpa_diagnostics'
get_results(x, what = NULL, ...)
```

```
## S3 method for class 'summary_multilpa'
get_results(x, what = NULL, ...)

## S3 method for class 'summary_multilpa_covariates'
get_results(x, what = NULL, ...)

## S3 method for class 'summary_multilpa_transitions'
get_results(x, what = NULL, ...)

## S3 method for class 'summary_multilpa_enumeration'
get_results(x, what = NULL, ...)

## S3 method for class 'summary_multilpa_bootstrap_lrt'
get_results(x, what = NULL, ...)
```

Arguments

x	A fitted model of this package, an enumeration grid, a bootstrap comparison, a starting-value set, a classification-diagnostics object, or the summary of any of them.
what	Which table to return, or "all" for every table the object can produce. The tables an object offers depend on its class and are listed under <i>Tables</i> below; asking for one it does not have raises <code>latents_bad_argument</code> naming the ones it does. NULL, the default, gives the object's primary table: the measurement model for a fit, the candidate grid for an enumeration, the test row for a bootstrap comparison, and the entropy summary for a diagnostics object.
...	Arguments for the requested table. Each table accepts only the arguments listed for it below, and any other name raises <code>latents_bad_argument</code> saying which that table takes, rather than being dropped on the way to a table that would then answer a different question.

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

Every fitted model offers these:

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

"responses" The categorical measurement model, one row per profile, categorical indicator and category: profile, indicator, category, probability, threshold. The threshold is `qlogis(P(y <= category))` and is `NA_real_` for each indicator's final category, where the cumulative probability is one. Takes `data`, which adds `probability_standard_error`.

- "covariances" The within-profile residual covariance matrices, one row per profile and ordered pair of continuous indicators: profile, indicator, indicator_2, covariance. The diagonal parameterization carries no covariance array but does state a covariance matrix – the variances on the diagonal and exact zeros off it – and it is reported explicitly, so one shape answers both parameterizations.
- "counts" Effective class memberships, one row per class at each level the fit has: level, class, effective_count (the summed posteriors) and effective_proportion. These differ from the modal counts in "classification" whenever entropy is below one.
- "model" One row describing the whole fit: its dimensions, likelihood, parameter counts, aic, bic_groups, bic_individual and its convergence and boundary diagnostics. The columns are those of the family: a covariate fit reports its predictor counts, because padding those onto a two-level fit would leave the common case mostly NA. Comparing fits across families is what "information_criteria" is for, and that table does have one column set for every family.
- "posteriors" One row per individual and profile: row, group, profile, posterior, modal. Takes format.
- "assignments" One row per observation, the columns the fit was built from (or those of data) followed by profile, group_class where the model has one, uncertainty, and one posterior_profile_* column per profile. uncertainty is one minus the posterior of the assigned profile, which is what modal assignment discards. Takes data and truth; truth returns a recovery cross-tabulation instead, described below.
- "classification" Classification quality, one row per class and level: level, class, n_modal, proportion_modal, estimated_n, estimated_proportion, average_posterior, odds_correct_classification. Takes level. Under sampling weights estimated_n and estimated_proportion (and so the odds of correct classification) are weighted posterior totals; the modal and average posterior columns describe the units supplied.
- "average_posteriors" For the units assigned to each class, their mean posterior of belonging to every class: level, assigned_class, class, n_assigned, average_posterior. Each assigned class's rows sum to one. Conditions on the *assigned* class, where "classification_errors" conditions on the *true* one; the two are different numbers, not two spellings of one table. Takes level.
- "classification_errors" The probability that a unit truly in one class is assigned to another: level, true_class, assigned_class, probability, summing to one within each true_class. Takes level. On a membership-covariate fit this is a marginal summary over the fitted units, not a classification-error adjustment conditional on covariates. A fit with a noise component raises latents_unsupported_noise, as [three_step\(\)](#) does, and "all" and summary() leave the table out.
- "bch_weights" The BCH inverse-error weights, one row per unit and class: level, unit, assigned_class, class, weight. Takes level; weight a regression with one level, not with "both". On a membership-covariate fit these invert the marginal error matrix and are not a conditional three-step correction. Refused for a fit with a noise component, like "classification_errors".
- "entropy" One row per level: level, n_classes, n_units, entropy_sum, relative_entropy.
- "residuals" Bivariate residuals for every indicator pair, testing the within-profile independence the measurement model assumes: profile, indicator_1, indicator_2, kind, observed, expected, residual, effective_n, statistic, df, p_value, p_adjusted. Takes data, by and adjust.

"information_criteria" Every likelihood-penalty criterion the package computes. Takes format and definitions.

"sequences" The assignments in occasion order, one row per observation: group, group_class, time, profile. Takes format. Raises latents_no_time for a fit made without time.

"sequence_summary" One row per group class, summarizing how much data it contributes: group_class, groups, observations, mean_length, median_length, shortest, longest, complete, gaps. 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.

"data" The columns of the original data the model was fitted to, one row per observation in input order: the identifier, the occasion where there is one, and the indicators under their original names. Columns a model never saw are not here, which is why `three_step()` and `r3step()` still take data.

Every family with discrete group classes adds:

"group_posteriors" One row per observed group and group class: group, group_size, log_likelihood, group_class, posterior, modal. Takes format.

And each family adds its own:

"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 a single "joint" row and a `fit_staged()` result has two; fixed is NA_character_ for a stage that estimated every block, never a sentinel such as "none".

"coefficients" A covariate fit. One row per estimated membership coefficient: level ("profile" or "group"), outcome, term, parameter (always "logit"), estimate. The standard errors are not here; `parameter_inference()` reports them.

"transitions", "initial", "sequence_lengths" A transition fit. "transitions" has one row per group class and ordered pair of profiles, with group_class, from, to, probability, expected_count, stable, estimated and group_class_probability; it takes estimated and stable to restrict it. "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", "criteria" An enumeration grid. "criteria" has one row per information criterion, naming the candidate that minimises it. "candidates" has one row per candidate model, with its class counts, its covariance model, log likelihood, parameter count, every criterion under both sample-size conventions, both entropies, and the convergence, boundary, replication, warning and error diagnostics. Failed candidates are retained with NA estimates and their error text.

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

"covariances" A starting-value set. One row per profile and ordered pair of continuous indicators: profile, indicator, indicator_2, covariance.

What "all" returns

A named list of every table the object can produce, in catalogue order, built from the same definitions a single what uses, so the two can never disagree. It takes no further arguments: every table is built with its defaults, and supplying anything else raises `latents_bad_argument`. A table this particular object cannot produce is left out rather than erroring – a sequence table for a fit made without time, bivariate residuals for a family with no discrete group classes – so the names of the list say what was available. Only the classed refusals that mean *this object has no such table* are skipped; anything else propagates.

Arguments the tables take

These reach the requested table through `...`, and each table takes only the ones listed for it under *Tables* above.

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. For "profiles" and "responses" it triggers `parameter_inference()` and adds a standard error beside every estimate.

scale For "profiles", "raw" reports the estimates in input units and "standardized" divides each indicator's deviation from its grand mean by that indicator's observed standard deviation, which is what `plot(x, scale = "standardized")` draws. Every standard error beside an estimate is divided by the same constant, so the whole row is on one scale.

format For the posterior tables and "sequences", "long" gives one row per unit and class and "wide" one row per unit with one column per class. For "information_criteria", "wide" is the one-row reporting shape and "long" gives one row per criterion and sample-size convention.

level Which level of classification to report: "individuals", "groups" or "both". NULL, the default, gives "both" for a fit that has discrete group classes and "individuals" for one that does not, which is the same rule `diagnostics()` follows.

truth For "assignments", one or more column names of data holding a known label to check the assignments against. See *Recovery* below.

by For "residuals", "profile" assesses each profile separately and "overall" pools them.

adjust For "residuals", the multiplicity correction applied across the indicator pairs.

definitions For "information_criteria" in long format, TRUE adds the formula and the reference for each criterion.

estimated, stable For "transitions", TRUE or FALSE restricts the table to the rows that were estimated from observed occupancy, or to the diagonal. NULL 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 (which model label the column was compared against), class, truth (the column name), value, n and proportion, the share of the units carrying that truth value that were assigned to that class. Several truth columns may be named at once and are stacked.

Each truth column is compared against the level it describes, which is read off the data rather than guessed: a column that takes one value within every group is a property of the group and is

compared against `group_class`; a column that varies inside any group cannot be a group-level label and is compared against profile. The assignment column records which comparison was made, so the choice is never silent. Group-level recovery counts each group once, even when groups have different numbers of observations. Unused factor levels are omitted, because their within-truth proportions are undefined. Every column is compared against profile for a fit with no discrete group classes.

The labels are the model's own and carry no order, so a recovery table whose mass sits off the diagonal is a relabelling, not a failure.

Information criteria

Let q be the number of free parameters, n the chosen sample size, and EN the classification entropy of the level matching that convention. The criteria are $\text{deviance} = -2L$, $\text{aic} = -2L + 2q$, $\text{bic} = -2L + q \log(n)$, $\text{sabic} = -2L + q \log((n + 2) / 24)$, $\text{caic} = -2L + q (\log(n) + 1)$, $\text{awe} = -2(L - EN) + 2q (1.5 + \log(n))$, $\text{icl} = -2L + q \log(n) + 2 EN$, $\text{kic} = -2L + 3(q + 1)$, and $\text{clc} = -2L + 2 EN$. Note that `clc` uses the entropy sum, as `icl` and `awe` here do; `tidyLPA` reports a CLC built from relative entropy instead, which is bounded by one and so penalizes almost nothing, and the two numbers are not comparable.

Every criterion that depends on a sample size is reported under both multilevel conventions, "groups" and "individuals", and uses group-level posteriors under the first and individual-level posteriors under the second. That is a stated per-level choice, not a unique multilevel definition: use one convention consistently across compared candidates. `deviance`, `aic` and `kic` depend on no sample size and carry `convention = NA_character_`. `clc` depends on none either, but its convention selects which level's classification uncertainty it penalizes, so it is reported once per convention. Individual sample sizes exclude rows with no observed indicators. Where the group classification entropy is undefined, group-level `awe`, `icl` and `clc` are `NA`.

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 proportion alone; values of five or more are conventionally read as adequate separation. It is `NA_real_` wherever it is undefined, that is whenever `average_posterior` is missing, or it or `estimated_proportion` is zero or one, so a single-class solution always reports `NA_real_`. `average_posterior` is itself `NA_real_` for a class with no modal members, where the mean is taken over nothing and is undefined rather than zero. Modal assignment discards classification uncertainty, which is why `n_modal` and `estimated_n` differ whenever entropy is below one.

`relative_entropy` is $1 - EN / (n \log K)$, on the zero-to-one scale, so higher is sharper; `entropy_sum` is the raw EN the entropy-penalized criteria use, so lower is sharper. `relative_entropy` is `NA_real_` when a level has a single class, where it is undefined rather than perfect.

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

Bivariate residuals

The measurement model assumes the indicators are independent within a profile. "residuals" tests that pair by pair, weighting every observation by the posterior the fit holds for it, so a frame

in any other row order would pair each observation with someone else's posterior; the alignment is checked rather than assumed. A large residual is evidence of local dependence between two indicators, which the enumeration grid will otherwise absorb by asking for an extra profile. The p-values are approximate, and `p_adjusted` applies the multiplicity correction named by `adjust` across the pairs.

References

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. 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., & Hagenars, 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.

See Also

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

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"))

# `engagement` and `student_type` are the kinds each row was simulated from,
# which the model never saw. Both recovery checks are one call.
get_results(fit, what = "assignments", data = course_engagement,
  truth = c("engagement", "student_type"))
```

```
get_results.latents_family_enumeration
```

Tables of a group-class family enumeration

Description

The result of `enumerate_classes(family = )` with group-class families: one fitted candidate per family, between-variance restriction and number of group classes.

Usage

```
## S3 method for class 'latents_family_enumeration'
get_results(
  x,
  what = c("candidates", "best"),
  criterion = c("bic", "aic", "bic_individual"),
  ...
)

## S3 method for class 'latents_family_enumeration'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

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

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

Arguments

<code>x</code>	A <code>latents_family_enumeration</code> result.
<code>what</code>	"candidates" (one row per candidate) or "best" (the candidate with the lowest value of criterion).
<code>criterion</code>	"bic" (penalized by the number of groups, the default), "aic" or "bic_individual".
<code>...</code>	Unused.
<code>row.names, optional</code>	Unused; part of the generic.
<code>object</code>	A <code>latents_family_enumeration</code> result.

Value

A base `data.frame`, one row per candidate: `model` (the code `candidate_fit()` takes), `family`, `between_variance`, `n_group_classes`, `log_likelihood`, `n_parameters`, `aic`, `bic`, `bic_individual`, `relative_entropy`, `smallest_class`, `min_effective_groups`, `converged`, `boundary`, `n_best_replicated`, and the candidate's warnings and error. With one group class every family is the same model, so those rows share a likelihood.

Examples

```

set.seed(1)
ratings <- data.frame(
  team = rep(seq_len(40), each = 6),
  climate = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
    rnorm(240))
grid <- enumerate_classes(ratings, "climate", "team",
  family = c("additive", "dispersion"),
  n_group_classes = 1:2, n_starts = 3, seed = 1)
get_results(grid)
get_results(grid, "best")

```

```
get_results.latents_growth_mixture
```

Tidy results of a growth mixture model

Description

Every table of a `mixture_regression()` fit with `random =` is a `base_data.frame` with one row per observation unit of that table.

Usage

```

## S3 method for class 'latents_growth_mixture'
get_results(
  x,
  what = "coefficients",
  level = 0.95,
  vcov_type = NULL,
  time = NULL,
  data = NULL,
  truth = NULL,
  ...
)

## S3 method for class 'latents_growth_mixture'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = "coefficients",
  ...
)

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

```

```
## S3 method for class 'latents_growth_mixture'
vcov(object, type = NULL, ...)

## S3 method for class 'latents_growth_mixture'
confint(object, parm, level = 0.95, ...)

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

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

Arguments

x	A latents_growth_mixture fit.
what	Which table: "coefficients" (one row per class and term, with Wald statistics; shared terms have class "common"), "classes" (share with interval, expected and assigned persons, average posterior, residual standard deviation), "random" (per class: random-effect variances, standard deviations and correlations with intervals formed on the log and Fisher-z scales), "membership" (membership logits against class 1), "fit" (one row: log likelihood, information criteria with persons as the sample size, entropy, convergence), "assignments" (one row per person: modal class and posteriors), "random_effects" (one row per person: predicted random effects under the modal class), "trajectories" (one row per class and time point: the mean trajectory with a confidence band), "individual" (one row per observation: observed value, class trajectory and the person's own predicted curve), "starts" (one row per start), "recovery" (needs data and truth: the modal classes cross-tabulated against a known classification that is constant within persons, with counts and shares of each assigned class), or "all" (a named list of every table but "recovery"). A multilevel growth mixture (cluster) adds "group_classes" (one row per group class and class: the class probabilities within the group class, their standard errors without membership covariates, the group class's share of the clusters and its assigned clusters) and "clusters" (one row per cluster: persons, modal group class and posteriors); its "membership" table holds the group-class logits (against group class 1) and the class logits with one intercept per group class, "assignments" gains each person's cluster, and "fit" counts clusters in bic (persons in bic_persons).
level	Confidence level of the intervals.
vcov_type	NULL (the stored type), "observed", "robust" or "opg". A weighted fit allows "robust" only.
time	For "trajectories" and "individual", the numeric variable to draw along; by default the first numeric variable of random.
data, truth	For "recovery": the data frame the model was fitted to and the name of its column holding the known classes.
...	Unused.

row.names, optional	Unused; for the generic.
object	A latents_growth_mixture fit.
type	As vcov_type.
parm	Unused.

Value

A base data.frame (or, for "all", a named list of them).

Examples

```
few_students <- subset(growth_scores, student <= 60)
small_fit <- mixture_regression(score ~ wave, few_students, n_classes = 2,
                              id = "student", class_level = "group",
                              random = "intercept", seed = 1)
get_results(small_fit, "classes")
get_results(small_fit, "random")

# All 300 students, with random intercepts and slopes
fit <- mixture_regression(score ~ wave, growth_scores, n_classes = 3,
                         id = "student", class_level = "group",
                         random = "wave", random_covariance = "equal",
                         n_starts = 3, seed = 1)
get_results(fit, "classes")
get_results(fit, "random")
get_results(fit, "recovery", data = growth_scores, truth = "trajectory")
```

get_results.latents_mixture_regression
Tables of a mixture-of-regressions fit

Description

Tables of a mixture-of-regressions fit

Usage

```
## S3 method for class 'latents_mixture_regression'
get_results(
  x,
  what = "coefficients",
  level = 0.95,
  vcov_type = NULL,
  data = NULL,
  truth = NULL,
  by = c("class", "group_class"),
```

```

    ...
)

## S3 method for class 'latents_mixture_regression'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = "coefficients",
  level = 0.95,
  vcov_type = NULL,
  ...
)

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

## S3 method for class 'latents_mixture_regression'
vcov(object, type = NULL, ...)

## S3 method for class 'latents_mixture_regression'
confint(object, parm, level = 0.95, ...)

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

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

```

Arguments

x	A fit from mixture_regression() .
what	The table to return; see <i>Tables</i> . "all" returns every table in a named list.
level	Confidence level for the intervals.
vcov_type	NULL for the covariance stored with the fit, or "observed", "robust" or "opg" to recompute it.
data, truth	For what = "recovery": the data frame the model was fitted to and the name of its column holding a known classification.
by	For what = "recovery": "class", or "group_class" for the group classes of a two-level fit.
...	Unused.
row.names, optional	Unused; part of the generic.
object	A fit from mixture_regression() .
type	Covariance type for <code>vcov()</code> ; as <code>vcov_type</code> .
parm	Parameter names or positions; all by default.

Value

A base data.frame as described under *Tables*, or a named list of them for what = "all".

Tables

- coefficients** One row per class and regression term: class ("common" for a shared term), term, estimate, std_error, statistic (Wald z), p_value, conf_low, conf_high, p_adjusted (Benjamini-Hochberg across the non-intercept rows; NA for intercepts) and, for the binomial and Poisson families, exp_estimate (odds or rate ratio). For the ordinal family each class's thresholds lead its rows as terms "threshold:<lower>|<upper>" (delta-method errors; no p_adjusted or exp_estimate), and exp_estimate of a slope is the cumulative odds ratio of a higher category.
- classes** One row per class: share (mean posterior), count (summed posterior), n_assigned (modal assignment), mean_posterior (average posterior of the units assigned to it); sigma and sigma_std_error for the Gaussian family; prior and prior_std_error when membership has no covariates. Units are groups under class_level = "group".
- membership** One row per multinomial-logit coefficient: model ("class" or "group_class"), class, term, reference, the Wald columns and odds_ratio. In the two-level model the class intercepts are per group class, (Intercept):group_class_h.
- group_classes** Two-level model: one row per group class and class, with the model-implied class probability, the group_share, n_groups_assigned, and std_error of the probability when membership has no covariates.
- fit** One row: log likelihood, parameter count, aic, bic (penalized by the number of independent units: rows, or groups when id defines them), bic_rows (always by rows, as flexmix reports it), sabic, icl, relative entropy (and group_entropy), the smallest class share, convergence and start replication.
- assignments** One row per data row used: row (its position in the supplied data), the id column, modal class, its posterior, and one probability_class_k column per class; two-level fits add group_class and group_posterior.
- groups** One row per group (class_level = "group" or the two-level model): its size, modal class or group class, and posterior probabilities.
- fitted** One row per data row: observed, the posterior-weighted fitted mean, fitted_modal (the modal class's mean), residual and class.
- starts** One row per start: its log likelihood, convergence, iterations, stage ("screened": stopped after the 50-iteration screening because better starts existed; "completed": run to convergence or max_iter), whether it degenerated, and which one was selected. n_best_replicated counts completed starts only.
- classification** One row per assigned (modal) class and class: the mean_posterior probability of class among the units assigned to assigned, and n_assigned. Its diagonal is each class's average posterior probability; values near one mean well-separated classes.
- recovery** Needs data and truth: the modal classes cross-tabulated against a known classification, one row per assigned class and true value, with the count n and its share of the assigned class. Group-level under class_level = "group" or by = "group_class", where truth must be constant within groups.

Examples

```
fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                          n_starts = 3, seed = 1)
get_results(fit, "coefficients")
get_results(fit, "fit")
get_results(fit, "coefficients", vcov_type = "robust")
get_results(fit, "recovery", data = study_hours, truth = "strategy")
```

get_results.latents_pooled

Tables of a pooled multiply imputed fit

Description

Tables of a pooled multiply imputed fit

Usage

```
## S3 method for class 'latents_pooled'
get_results(x, what = c("estimates", "imputations", "fits"), ...)

## S3 method for class 'latents_pooled'
as.data.frame(x, row.names = NULL, optional = FALSE, what = "estimates", ...)
```

Arguments

x	A latents_pooled object from pool_imputations() .
what	"estimates" (the pooled table), "imputations" (every imputation's aligned estimates) or "fits" (one row per imputation).
...	Unused.
row.names, optional	Unused; part of the generic.

Value

A base data.frame; see [pool_imputations\(\)](#) for the columns.

Examples

```
set.seed(1)
# Two stand-in completed data sets. With real missing covariates, pass the
# `mids` object mice::mice() returns instead.
completed <- lapply(1:2, 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)
get_results(pooled, "imputations")

```

```
get_results.latents_transition_enumeration
```

Tables of a transition-model enumeration

Description

The result of `enumerate_classes(..., time = )`: one transition model per number of profiles and group classes (and covariance structure, when model names several).

Usage

```

## S3 method for class 'latents_transition_enumeration'
get_results(
  x,
  what = c("candidates", "best"),
  criterion = c("bic", "aic", "bic_individual"),
  ...
)

## S3 method for class 'latents_transition_enumeration'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

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

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

```

Arguments

<code>x</code>	A <code>latents_transition_enumeration</code> result.
<code>what</code>	"candidates" (one row per candidate) or "best" (the lowest value of criterion).
<code>criterion</code>	"bic" (penalized by the number of groups), "aic" or "bic_individual".
<code>...</code>	Unused.
<code>row.names, optional</code>	Unused; part of the generic.
<code>object</code>	A <code>latents_transition_enumeration</code> result.

Value

A base `data.frame`, one row per candidate: `n_profiles`, `n_group_classes`, `model`, `log_likelihood`, `n_parameters`, `aic`, `bic`, `bic_individual`, `relative profile entropy`, `converged`, `boundary`, `n_best_replicated`, `warnings` and `error`.

Examples

```
# The first 50 students keep the example quick
few_students <- subset(course_engagement, student <= 50)
grid <- enumerate_classes(few_students, c("browse", "lectures"),
                          "student", time = "sequence", n_profiles = 2:3,
                          n_group_classes = 1, n_starts = 2, seed = 1)

get_results(grid)
get_results(grid, "best")
```

```
get_results.mutilpa_additive
```

Tables of an additive group-class fit

Description

Tidy tables of a fit from `multilpa(family = "additive")`. Every table is a `base::data.frame`; the fit itself never needs to be reached into.

Usage

```
## S3 method for class 'multilpa_additive'
get_results(
  x,
  what = "parameters",
  level = 0.95,
  vcov_type = c("observed", "robust", "opg"),
  adjust = .multilpa_p_adjust_methods,
  data = NULL,
  truth = NULL,
  ...
)

## S3 method for class 'multilpa_additive'
as.data.frame(x, row.names = NULL, optional = FALSE, what = "parameters", ...)

## S3 method for class 'multilpa_additive'
coef(object, scale = c("natural", "unconstrained"), ...)

## S3 method for class 'multilpa_additive'
vcov(
  object,
  type = c("observed", "robust", "opg"),
  scale = c("natural", "unconstrained"),
  ...
)

## S3 method for class 'multilpa_additive'
```

```

confint(object, parm, level = 0.95, type = c("observed", "robust", "opg"), ...)

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

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

```

Arguments

<code>x</code>	A <code>mutilpa_additive</code> fit.
<code>what</code>	Which table; "all" returns a named list of every table.
<code>level</code>	Confidence level of the intervals in "parameters".
<code>vcov_type</code>	"observed" (inverse observed information), "robust" (CR0 sandwich over groups) or "opg" (inverse outer product of the group scores).
<code>adjust</code>	p-value adjustment across the class means, one of stats::p.adjust.methods ; "none" by default, as elsewhere in <code>latents</code> .
<code>data, truth</code>	For <code>what = "recovery"</code> only: the data the fit was made from and the name of a column, constant within groups, holding a known group classification.
<code>...</code>	Unused.
<code>row.names, optional</code>	Unused; part of the generic.
<code>object</code>	A <code>mutilpa_additive</code> fit.
<code>scale</code>	"natural" (means, variances, class weights) or "unconstrained" (means, log variances, class-1-baseline logits). Names of <code>coef()</code> and the rows of <code>vcov()</code> agree on either scale.
<code>type</code>	Covariance type, as <code>vcov_type</code> .
<code>parm</code>	Natural parameter names or positions; all by default.

Value

A base `data.frame`, or a named list of them for `what = "all"`.

Tables

`parameters` One row per natural parameter: `level` ("between" for class means and between-group variances, "within" for the shared within-group variances, "group_class" for class weights), `group_class` ("shared" for a parameter common to all classes), `indicator` (NA for weights), `parameter` ("mean", "variance", "weight"), `estimate`, `standard_error`, `statistic`, `p_value`, `p_adjusted`, `conf_low`, `conf_high`. Only means carry a Wald test. Intervals for variances are formed on the log scale and for weights on the logit scale. When inference is refused (boundary, unconverged, singular information) the standard-error columns are NA and a `latents_no_standard_errors` message says why.

`group_classes` One row per group class: `weight`, `count` (summed posterior), `n_assigned` (modal), `mean_posterior` among the groups assigned to it, and `effective_groups`: the number of perfectly classified groups that would estimate the class weight as precisely as this fit does,

$w^2 (1 - w) / \text{Var}(w)$. It equals the class's expected group count when classes are perfectly separated and falls when a class is small or poorly separated. Below 50, intervals for class parameters are flagged with a `latents_weak_class` warning (threshold and its simulation evaluation: `validation/ADDITIVE_SIMULATION.md` in the source repository). NA for one class or when inference is refused.

`groups` One row per group (named by the `id` column): size `n`, modal `group_class`, its posterior, and `probability_group_class_h`.

`intercepts` One row per group and indicator: `n`, the observed group average, and the posterior mean and variance of the group's intercept, mixed over its class posterior (so the variance includes uncertainty about class membership). Conditional on the estimated parameters; not a statement about their sampling uncertainty.

`conditional_intercepts` As `intercepts`, one row per group, indicator and group class, conditional on that class, with the class posterior.

`assignments` One row per data row: `row`, the `id` column, the group's modal `group_class` and posterior, and `assignment_level = "group"`: every row inherits its group's class.

`classification` One row per assigned and group class: the `mean_posterior` of `group_class` among groups assigned to assigned.

`fit` One row: log likelihood, `n_parameters`, `aic`, `bic` (penalized by the number of groups), `bic_individual` (by rows), `n_groups`, `n_obs`, relative entropy over groups, the smallest class weight, `min_effective_groups`, `converged`, `boundary`, `kkt` and start replication.

`recovery` Needs data and truth: one row per modal group class and known value, with the number of groups `n` and their share of the assigned class. Not part of what = "all".

`starts` One row per start: likelihood, convergence, iterations, number of between variances held at zero, error message of a failed start, and which start was selected.

Examples

```
set.seed(1)
ratings <- data.frame(
  team = rep(seq_len(40), each = 6),
  climate = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
    rnorm(240),
  support = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
    rnorm(240))
fit <- multilpa(ratings, c("climate", "support"), "team",
  n_group_classes = 2, family = "additive", n_starts = 3,
  seed = 1)
get_results(fit)
get_results(fit, "group_classes")
get_results(fit, "parameters", vcov_type = "robust")
```

get_results.multipa_cross_level

Tables of a cross-level fit

Description

Tidy tables of a fit from `mutilpa(family = "restricted_cross_level")` or `mutilpa(family = "full_cross_level")`.

Usage

```
## S3 method for class 'mutilpa_cross_level'
get_results(x, what = "profiles", ...)

## S3 method for class 'mutilpa_cross_level'
as.data.frame(x, row.names = NULL, optional = FALSE, what = "profiles", ...)

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

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

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

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

Arguments

<code>x</code>	A <code>mutilpa_cross_level</code> fit.
<code>what</code>	Which table; "all" returns a named list of every table.
<code>...</code>	Unused.
<code>row.names, optional</code>	Unused; part of the generic.
<code>object</code>	A <code>mutilpa_cross_level</code> fit.

Value

A base `data.frame`, or a named list of them for `what = "all"`.

Tables

`profiles` One row per individual profile and indicator: `level = "individual"`, `profile`, `indicator`, `mean`, `variance`; then one row per group class and indicator for the group means: `level = "group"`, the class in profile, and the class's mean and variance of the group means.

`group_classes` One row per group class: `weight`, `count` (summed posterior) and `n_assigned`.

`composition` One row per group class and profile: the share of the class's members in each profile. Model-implied for the full family; for the restricted family a posterior-weighted description, since its prevalences do not depend on the class.

`groups` One row per group: size, modal group class, its posterior and every class probability.

assignments One row per data row: its modal profile and posterior, and its group's modal group_class (inherited).

fit One row: family, log likelihood, n_parameters, aic, bic (by groups), bic_individual (by rows), sizes, convergence and boundary. The likelihood is the working likelihood of the manifest specification (the group means are computed from the same ratings), so it is comparable only between cross-level fits of the same data.

starts The start records (full family) or the two parts' likelihoods (restricted family).

Examples

```
set.seed(1)
ratings <- data.frame(
  team = rep(seq_len(40), each = 6),
  climate = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
    rnorm(240, sample(c(-1, 1), 240, replace = TRUE), 0.6))
fit <- mutilpa(ratings, "climate", "team", n_profiles = 2,
  n_group_classes = 2, family = "full_cross_level",
  n_starts = 3, seed = 1)
get_results(fit)
get_results(fit, "composition")
```

get_results.mutilpa_lta

Tables of a latent transition fit with occasion- or covariate-dependent transitions or occasion-specific measurement

Description

The result of `lta()` with `transitions = "occasion"`, `transition_covariates`, `initial_covariates` or `measurement = "occasion"`.

Usage

```
## S3 method for class 'mutilpa_lta'
get_results(
  x,
  what = "transitions",
  level = 0.95,
  vcov_type = c("observed", "robust", "opg"),
  ...
)

## S3 method for class 'mutilpa_lta'
as.data.frame(x, row.names = NULL, optional = FALSE, what = "transitions", ...)

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

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

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

## S3 method for class 'multilpa_lta'
vcov(object, type = c("observed", "robust", "opg"), ...)

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

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

Arguments

<code>x</code>	A multilpa_lta fit.
<code>what</code>	Which table; "all" for a named list of every table.
<code>level</code>	Confidence level of the coefficient intervals.
<code>vcov_type</code>	"observed", "robust" (CR0 over groups) or "opg".
<code>...</code>	Unused.
<code>row.names, optional</code>	Unused; part of the generic.
<code>object</code>	A multilpa_lta fit.
<code>type</code>	Covariance type, as <code>vcov_type</code> .

Value

A base data.frame, or a named list of them for `what = "all"`.

Tables

`transitions` Model-implied transition probabilities, one row per group class, origin from and destination to; with occasion-varying or covariate-dependent transitions, one set per destination occasion, averaged over the groups observed there.

`transition_coefficients` One row per group class, move from -> to and design term: the log odds of that move rather than staying in from, its standard error, Wald statistic, p-value, interval and odds_ratio. Terms are (Intercept) or occasion_t and each covariate column.

`second_order_transitions, second_order_coefficients` With `order = 2`: the probability of moving to given the profiles at the two previous occasions (previous, from), averaged over the groups observed from the third occasion on, and its logit coefficients (reference: staying in from). The transitions tables then describe the first move only.

`initial, initial_coefficients` The initial distribution (averaged over groups) and its logit coefficients, reference: the last profile.

profiles Means and variances per profile and indicator, per occasion when measurement is occasion-specific.

group_classes, assignments, fit, starts As for other fits.

Examples

```
# Does the grade in the previous course shift the chance of switching
# engagement profile? `previous_grade` changes from course to course.
fit <- lta(course_engagement, c("browse", "lectures"), "student",
          n_profiles = 2, time = "sequence",
          transition_covariates = "previous_grade", n_starts = 2, seed = 1)
get_results(fit, "transition_coefficients")
get_results(fit, "transitions")
```

get_tna	<i>A transition network for the whole sample</i>
---------	--

Description

Builds one `tna::tna()` model from a fitted transition model, aggregated over every latent group class.

Usage

```
get_tna(x, ...)

## S3 method for class 'multilpa_transitions'
get_tna(x, ...)

## S3 method for class 'multilpa_lta'
get_tna(x, ...)

## S3 method for class 'multilpa'
get_tna(x, ...)

## S3 method for class 'multilpa_covariates'
get_tna(x, ...)
```

Arguments

x	A fitted model from <code>lta()</code> .
...	Passed to <code>tna::tna()</code> .

Details

The aggregate is formed from the expected transition counts, summed across classes and then normalised, rather than by averaging the class transition matrices. The two differ: averaging weights each class by how probable it is, and summing counts weights it by how much transition mass it actually contributes, which is what a marginal transition probability means. A class holding a tenth of the students but a fifth of the observed moves counts for the latter. If a profile has no expected outgoing moves in any class, its transition row has no count-based estimate; the network uses the fitted class rows averaged by class probability and warns that the row is unestimated. It never interprets absent moves as a certain self-transition.

A general `lta()` fit – with occasion-varying or covariate-dependent transitions, a second order, movers and stayers, or ordinal, count or weighted data – has no single transition matrix, and is refused with `latents_unsupported_tna`; its transitions are in `get_results(x, "transitions")`.

For a `multilpa()` or `multilca()` fit made with `time =`, there is no estimated transition matrix: the network is built by `tna::tna()` from each group's sequence of modal profiles in time order, the observed transitions between the profiles the observations were assigned to. `get_group_tna()` splits those sequences by each group's modal group class. A fit made without `time` has no order and is refused.

Value

An object of class `tna`, as `tna::tna()` returns: every verb of that package applies to it, including `centralities()`, `communities()`, `cliques()` and `plot()`.

See Also

`get_group_tna()` for one network per latent 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

A simulated multilevel longitudinal dataset made for the multilevel growth mixture model. 600 students in 40 schools (15 per school) sit a reading test on five occasions (wave 0 to 4). Two kinds of student follow two trajectories: *improving* (starts near 48, gains 3 points a wave) and *plateauing* (starts near 56, gains 0.3). Students vary around their trajectory (random intercept SD 3, random slope SD 0.6, correlation -0.2; residual SD 2). Schools are of two types: in *supportive*

schools 80% of students improve, in *struggling* schools 25% do, and a school's support programme (standardized, measured once) makes it more likely to be supportive. About 8% of tests were missed at random.

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 type. Not for fitting; kept to check recovery.

trajectory Factor, "improving" or "plateauing": the student's generating kind. Not for fitting; kept to check recovery.

Details

The school types are what a multilevel growth mixture model's group classes recover: `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.latents_growth_mixture()`.

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

Achievement growth of students with three kinds of trajectory

Description

A simulated longitudinal dataset made for growth models. 300 students sit an achievement test on up to seven occasions (wave 0 to 6). Three kinds of student follow three trajectories: *improving* (starts near 48, gains 3.5 points a wave), *stable* (starts near 62, gains 0.5) and *declining* (starts near 58, loses 2.5). Within each kind, students vary around the trajectory: each has their own start and rate of change (random intercept SD 4, random slope SD 0.8, correlation -0.3, the same spread in every kind), with residual SD 3. Motivation, measured once, makes a student more likely to be improving and less likely to be declining. About 12% of tests were missed at random, so the panel is unbalanced.

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 kind. Not for fitting; kept to check recovery.

Details

Because students vary within their kind, a trajectory model without random effects needs more classes than the three that generated the data; a growth mixture model with random intercepts and slopes recovers three.

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	<i>Conditions this package raises</i>
--------------------	---------------------------------------

Description

Every error a caller might reasonably want to handle carries a class, so it can be caught by what went wrong rather than by matching message text, which changes between versions. This page is the contract: the classes below are stable, the messages are not.

Value

Nothing; this page documents conditions rather than a function.

Input and data contracts

- `latents_bad_data` The data frame, the indicator columns or the group column break the input contract: too few rows, a duplicated or absent column name, an indicator that is not a numeric vector, an indicator that is constant or wholly unobserved, missing or non-finite values under missing = "error", or a missing group identifier.
- `latents_bad_argument` A scalar argument is outside its contract: `n_profiles`, `n_group_classes`, `n_starts`, `max_iter`, `tol`, `min_variance`, or a seed or seeds that is not a whole number set.seed() accepts.
- `latents_bad_time` time is missing, or repeats a position within a group, so the ordering it is supposed to supply does not exist.
- `latents_bad_categorical` A categorical indicator cannot be encoded, or `min_probability` leaves no room for its number of categories.
- `latents_bad_start` Supplied starting values are missing a block the model needs, do not have the shape it requires, or carry labels that name an indicator or a category this fit does not have.
- `latents_bad_fixed` A fixed request names a block the model does not have, or one that start does not supply, or start is absent.
- `latents_bad_stage` A measurement fit offered to `fit_staged()` has already estimated multiple group classes, or was fitted to different indicators, profiles or measurement options than the stage requested.
- `latents_bad_transition` A transition model was asked for with fewer than two profiles, or with no group observed at two occasions.

- `latents_bad_inference_data` The data passed to a verb alongside a fit do not reproduce the fit: a different number of rows, different values, or a different group ordering. Raised for a plain multilpa fit and for a covariate fit alike.
- `latents_no_indicator_data` A fit carries no stored indicators, so the request cannot be answered without the original data.
- `latents_incomplete_fit` A fit is missing fields the request needs, usually because it was made by an older version. Raised instead of letting a list subset by an absent name yield a NULL element keyed NA, which would leave a hole in the result.

Estimation and identification

- `latents_unidentified` The requested model cannot be identified from these 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.
- `latents_all_starts_failed` Every EM start raised, so no fit was produced. The message carries 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 the profile prevalence within it is not defined.
- `latents_too_few_groups` Fewer independent groups than the quantities being reported, so the covariance cannot be formed. Raised by robust `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` Classes cannot be separated well enough for the requested three-step correction.
- `latents_no_converge` A fit required by the request did not converge, and using it would report an unconverged estimate as final. Raised for a plain multilpa fit and for a covariate fit alike.
- `latents_boundary_fit` A variance or a response probability sits at its bound, or a mixing probability is zero, so the parameter is on the edge of its space and the asymptotic normal approximation Wald inference relies on does not hold. The standard error is refused rather than reported.
- `latents_singular_information` The observed information is not positive definite, or is numerically singular, so it cannot be inverted and no standard error can be formed.
- `latents_unknown_candidate` The requested profile and group-class combination was never enumerated.
- `latents_failed_candidate` An enumerated candidate could not be fitted. The message carries that candidate's error.
- `latents_no_group_classes` A verb that needs discrete group classes was called on a model that has none. Every model family this version fits has them, so this is a guard against a future family rather than a refusal the current surface can produce.

Model comparison

- `latents_bad_nesting` Two models are not nested in the way 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 or centering mode where the comparison requires them to match.
- `latents_reversed_likelihood` The model with more parameters has the lower likelihood, so at least one fit is at a local optimum and the comparison is meaningless.
- `latents_bad_outcome` A distal outcome given to `three_step()` is not of a type the requested method can handle, varies within a group when a group-level outcome is requested, or is a measurement indicator that already helped define the classes.
- `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 that already helped define the classes. The counterpart of `latents_bad_outcome`.
- `multilpa_indicator_reused` A measurement indicator was offered as an external variable, to `three_step()` as an outcome or to `r3step()` as a predictor. Raised alongside `latents_bad_outcome` or `latents_bad_covariate`, so one handler catches the mistake from either verb.
- `latents_bad_scores` Score contributions could not be formed for the robust sandwich.

Unsupported requests

These name capabilities the package does not have. They are raised in place of returning a number that would be wrong.

- `latents_unsupported_inference` Standard errors are not available for this particular fit. Raised for a covariate fit with categorical indicators, where no score is implemented for the response probabilities, for fits made by an older version, for a fit recording a covariance structure code this version does not know, and by `parameter_inference(method = "bootstrap")` for a fit holding a measurement block, whose held values came from a fit these data do not resample, and by `parameter_inference()` for a fit made with prior, which sits at a posterior mode rather than a likelihood maximum.
- `latents_unsupported_prior` `multilpa(prior = )` was asked for a combination 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 asked for more than one group class, categorical indicators, a fixed block, membership covariates or missing indicator values; or a verb that does not yet account for a noise component — `parameter_inference()`, `three_step()`, `r3step()`, `bootstrap_lrt()`, `starting_values()`, bivariate residuals, `fit_staged()` or a posterior plot — was given a fit that has one.
- `latents_unsupported_structure` `multilpa()` was asked for a covariance structure other than EEI, VVI, EEE or VVV together with `profile_covariates` or `group_covariates`, which fit those four only.
- `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 verb or option that does not account for sampling weights: `bootstrap_lrt()`, `enumerate_regressions(bootstrap = )`, `three_step()`, `r3step()`, prior or noise; or a non-sandwich `vcov_type` was asked of one.
- `latents_unsupported_indicator` ordinal or count indicators were combined with an option that does not take them yet: membership covariates, start, fixed, prior or noise.

- `latents_pooling_failed` An imputation's fit or inference failed in `pool_imputations()`, or the imputations' fits report different parameters. Rubin's rules need every imputation, so none is dropped; the message names the imputation and carries the original reason.
- `latents_bootstrap_failed` Fewer than two resamples produced a usable fit, so there is nothing to read a spread from. Raised by `parameter_inference(method = "bootstrap")`; the message carries the first reason a resample gave.
- `latents_no_free_parameters` The fit holds every parameter it has, so there is no free coordinate to report a standard error for. Raised by `parameter_inference()` and `vcov()` on a fully held fit.
- `latents_held_parameter` A named parameter was held fixed by this fit, so it has no sampling distribution and no confidence interval. Raised by `confint()` when `parm` names a held coordinate.
- `latents_unsupported_sensitivity` `sensitivity()` was called on a model family whose refit needs arguments the shared refit does not carry, whose profile labels cannot yet be aligned between two fits, or a directly fixed fit whose original free starts cannot be replayed.
- `latents_unsupported_three_step` `three_step()` or `r3step()` was called on a membership-covariate fit. Its classification errors depend on the fitted predictors, but the correction here uses one unconditional error matrix.
- `latents_unsupported_bootstrap` `bootstrap_lrt()` was called on a person-centred fit. Group baselines were removed before estimation and no distribution for them is available to simulate raw data under the null model.
- `latents_unsupported_tna` `get_tna()` or `get_group_tna()` was called on a general transition fit (a `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 to hand over; its transitions are in `get_results(x, "transitions")`.
- `latents_no_plot` `plot(x, what = "all")` was called on an object whose plot method names no views to draw.
- `latents_no_time` A sequence verb was called on a fit made without `time`, so it carries no ordering.
- `latents_missing_package` A verb needs a package listed in `Suggests` that is not installed. Raised by `get_tna()` and `get_group_tna()` when the `tna` package 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 this fit does not have.
- `latents_nothing_to_describe` `descriptives()` was given no variables to describe: either `vars` named `none`, or the frame has no numeric column outside `id` and by to fall back on.
- `latents_nothing_to_plot` The requested panel has no rows to draw.
- `latents_unknown_category`, `latents_unknown_criterion` A named category or information criterion does not exist in this fit.
- `latents_bad_scale` An unrecognised plotting scale was requested.

Warnings

A fit that is returned but qualified warns rather than errors, so that a simulation loop can catch the qualification it cares about and let the others through.

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

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

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

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

`latents_separation` Warning. 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 roundoff. This indicates a defect and should be reported.

`latents_unconverged` The best start had not converged when `max_iter` was reached, so the returned estimate is not a maximum.

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

`latents_small_classes` A profile or group class has effective membership below one, so it is supported by less than one observation.

`latents_weak_class` `multilpa(family = "additive")` only: a group class is supported by fewer than 50 effective groups (see `effective_groups` in `get_results.multilpa_additive()`), because it is small, poorly separated, or both. Standard errors are still reported, but intervals for class means and weights can 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 left out of the interval. Raised by `parameter_inference(method = "bootstrap")`, naming how many, so the count the interval rests on is never quietly smaller than `iter`.

`latents_sensitivity_dropped` One or more seeds did not produce a fit in `sensitivity()`. Their rows are NA rather than absent, and the warning names how many failed and the first reason, so the table is never quietly shorter than seeds.

latents_extreme_coefficients A membership logit coefficient is large enough that the class is close to separated, so the estimate is driven by scaling, a sparse class or separation rather than by the data.

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 them. The transition-network handoff also warns when it includes such a row, because it cannot be read as evidence of persistence.

latents_unverified_alignment A supplied data frame has too little fitted information to establish row order, for example only a repeated group identifier. The result is returned on the caller's assurance that the rows are in fitting order.

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)
# A sequence verb on a fit made without `time` is catchable by class.
tryCatch(get_results(fit, "sequences"),
  latents_no_time = function(condition) {
    "this fit carries no ordering"
  })
```

lca

Latent class analysis

Description

Fits a single-level latent class model: each observation belongs to one of `n_classes` unobserved classes, and within a class the discrete indicators are independent. Categorical indicators have unrestricted response probabilities; ordinal and count in `...` choose the structured measurement models described by `multilpa()`. It is the model `multilca()` fits with `id = NULL`, and it returns the same object, so every accessor, plot and inference verb of the package applies to it. In the results the classes are labelled `profile_1`, `profile_2`, and so on, as throughout the package.

Usage

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

Arguments

<code>data</code>	A data frame with one row per observation.
<code>vars</code>	Names of the discrete indicator columns. Categories are taken from the data; ordinal and count in <code>...</code> identify structured discrete indicators and the remaining columns are categorical.

`n_classes` Number of classes, a positive whole number.
`...` Further arguments for `multilca()`, such as `missing`, `n_starts`, `seed`, `min_probability` and `profile_covariates`, except `id` and `n_group_classes`, which a single-level model does not have.

Details

For observations nested in groups, use `multilca()` with the grouping column as `id`, which adds latent classes of groups above the classes of observations.

Value

A fitted `multilpa` object, as `multilca()` returns.

Conditions

`latents_bad_argument` when `id` or `n_group_classes` 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	<i>Latent profile analysis</i>
-----	--------------------------------

Description

Fits a single-level latent profile model: a finite mixture of multivariate normal distributions, in which each observation belongs to one of `n_profiles` unobserved profiles and each profile has its own mean on every indicator. It is the model `multilpa()` fits with `id = NULL`, and it returns the same object, so every accessor, plot and inference verb of the package applies to it.

Usage

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

Arguments

<code>data</code>	A data frame with one row per observation.
<code>vars</code>	Names of the indicator columns. Numeric columns are continuous indicators; name categorical ones in <code>categorical</code> .
<code>n_profiles</code>	Number of profiles, a positive whole number.
<code>...</code>	Further arguments for <code>multilpa()</code> : the covariance structure (<code>variance_model</code> , <code>covariance_model</code> , or <code>volume</code> , <code>shape</code> , <code>orientation</code>), <code>missing</code> , <code>categorical</code> , <code>n_starts</code> , <code>seed</code> , <code>prior</code> , <code>noise</code> , <code>profile_covariates</code> and the others it documents, except <code>id</code> and <code>n_group_classes</code> , which a single-level model does not have.

Details

For observations nested in groups (students in schools, occasions in persons), use `multilpa()` with the grouping column as `id`, which adds latent classes of groups above the profiles.

Value

A fitted `multilpa` object, as `multilpa()` returns.

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")
```

Description

Fits latent profiles to repeated observations of the same group and estimates the probabilities of moving between them from one occasion to the next. The measurement model is that of [multilpa\(\)](#) – Gaussian, categorical or mixed indicators, invariant across occasions – so profiles keep the same meaning at every occasion and a change of profile is a change of state rather than a change of definition. With more than one group class, each class carries its own initial distribution and its own transition matrix, which is how groups that differ in their dynamics are separated from groups that differ only in where they start.

Usage

```
lta(
  data,
  vars,
  id,
  n_profiles,
  time,
  n_group_classes = 1L,
  variance_model = c("varying", "equal"),
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  min_variance = 1e-06,
  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")
)
```

Arguments

<code>data</code>	A data frame containing indicators and a group identifier.
<code>vars</code>	Unique character vector of continuous indicator column names.
<code>id</code>	Name of the observed group identifier column. Character, factor, or numeric identifiers are supported; missing identifiers are not. Required here: this model has a second level by construction, so it has no single-level form and does not take <code>multilpa()</code> 's <code>id = NULL</code> .
<code>n_profiles</code>	Integer number of individual profiles, at least two.
<code>time</code>	Name of the column giving each observation's occasion within its group. Required: this model is defined by the ordering. Values must be complete and unique within each group.
<code>n_group_classes</code>	Positive integer number of latent group classes. One gives an ordinary single-level latent transition model; more than one fits a mixture of transition models over groups.
<code>variance_model</code>	Either "varying" (profile-specific variances or covariance matrices) or "equal" (shared across profiles).
<code>n_starts</code>	Positive integer number of EM starts. The first is deterministic: Ward's hierarchical clustering of the standardized indicators (on at most 2000 evenly spaced rows), each cluster described by its own means and spread, which is the start <code>mclust</code> 's name refers to and finds solutions whose profiles differ in shape. The rest are k-means from random centres. When <code>start</code> is supplied, it supplies the first start instead; remaining starts are random initializations. It is ignored when <code>max_iter = 0</code> and <code>start</code> is supplied: that call evaluates the supplied parameters and nothing else, so exactly one start is run whatever <code>n_starts</code> says.
<code>max_iter</code>	Nonnegative integer maximum number of EM updates per start. <code>max_iter = 0</code> performs no update. With <code>start</code> , it returns the model evaluated at exactly those values, whatever <code>n_starts</code> is, so that <code>logLik()</code> scores a parameter set supplied from elsewhere rather than the best of some random initializations that were never asked for. Without <code>start</code> there is nothing to evaluate at, so the random initializations are scored and the highest is returned.
<code>tol</code>	Positive relative log-likelihood tolerance. Convergence requires absolute change no greater than <code>tol * (1 + abs(previous log likelihood))</code> .
<code>min_variance</code>	Positive lower bound on each variance, or each covariance eigenvalue for full covariance, in squared input units. This defines a constrained maximum-likelihood problem. Bound-active estimates are explicitly reported and generate a warning.
<code>seed</code>	Optional random seed: any whole number set. <code>seed()</code> accepts, negative ones included. With a supplied seed, the caller's random-number state is restored on exit.
<code>missing</code>	"error" rejects missing indicators; "fiml" maximizes the observed-data likelihood under an ignorable missingness mechanism (MAR). Missing indicators are integrated out, not filled in for likelihood fitting. This holds with membership covariates too; the covariates themselves must be complete (<code>latents_bad_data</code>).

covariance_model	"diagonal" 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, which is the latent class measurement model. Binary, ordinal and unordered indicators are all handled by the same unrestricted 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, defining a constrained maximum-likelihood problem in the same way min_variance does for Gaussian indicators.
occasions	How a group's occasions are placed on the transition grid. "observed" 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 consumes a transition across the gap and contributes no measurement information at it. The two agree whenever every group is observed at every position.
select_start	Which start the fit reports. "likelihood", the default, takes the highest log likelihood across starts, preferring a converged start only among those tied with it to within a relative 1e-10. "converged" takes the highest log likelihood among the converged starts whenever at least one converged, and falls back to every start when none did. Use it when an unconverged start edges out a converged one by a negligible amount, which happens when a membership logit creeps along a flat ridge of the likelihood: the converged start is then a maximum that the standard errors can work with, and the other is not. summary() and get_results(fit, "starts") show every start either way.
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, ...).
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.
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.
measurement	"invariant" (the default) keeps the profiles' means, variances and response probabilities equal across occasions; "occasion" estimates them separately at each occasion, so a profile is defined by its position in the sequence of transitions rather than by one fixed measurement model.
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 move stays first order.
model	A covariance structure as multilpa() takes it ("EEI", "VVI", "VEI", ... ; mclust's codes). Standard errors are given for EEI, VVI, EEE and VVV in

	the homogeneous model and for the diagonal EEI and VVI with the extensions; other structures are refused (<code>latents_unsupported_inference</code>).
<code>mover_stayer</code>	TRUE adds a class of stayers: groups that never change profile (identity transitions) and have their own initial profile distribution (Goodman's mover-stayer model). <code>n_group_classes</code> then counts the mover classes; the stayer class is reported as "stayers".
<code>weights</code>	NULL, or the name of a column of data holding one sampling weight per id (constant over its occasions). As in <code>multilpa()</code> : 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 <code>multilpa_lta</code> .
<code>ordinal</code> , <code>count</code> , <code>count_model</code> , <code>count_dispersion</code>	Ordinal and count indicators, as in <code>multilpa()</code> ; with <code>measurement = "occasion"</code> their parameters are estimated per occasion. Read them with <code>get_results(fit, "ordinal")</code> and <code>get_results(fit, "count_means")</code> . A fit with them uses the general transition engine.

Details

This is latent transition analysis (LTA); with more than one group class it is a mixture over transition patterns, so the classes are trajectories rather than states. `multilpa()` is the cross-sectional counterpart.

By default transitions are first order and homogeneous over occasions: the probability of moving from one profile to another does not depend on the occasion or on earlier profiles. 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 rather than estimated. Such a row is warned about 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.

`max_iter = 0` updates nothing: the starting values are evaluated and returned with the expectation they imply, and the fit reports `converged = FALSE` after zero iterations. Every reported quantity, including `expected_count` and the implied profile prevalence, is then read off that single expectation.

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. Use `get_results()` for the tidy transition table and for the other tables. No standard errors, likelihood-ratio tests or guarantees of global optimality are given for this model family.

With `transitions = "occasion"`, `transition_covariates`, `initial_covariates`, `measurement = "occasion"` or `order = 2` the result is a `multilpa_lta` object instead, read with `get_results.multilpa_lta()`: transition and initial logit coefficients with Wald standard errors (observed, robust or OPG, from

analytic scores), model-implied transition probabilities per occasion, profiles per occasion. 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. Agreement with Mplus (User's Guide examples 8.13, 8.14; occasion-specific thresholds), depmixS4 and LMest is recorded in `equivalence/lta-extensions/` of the source repository.

References

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

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

```
# Students take their courses in their own order, which `sequence` records,
# so the same enrolments that fit a cross-sectional model fit a transition
# one. Engagement mostly persists from one course to the next.
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	<i>Fit a finite mixture of regressions</i>
--------------------	--

Description

Fits a regression whose coefficients differ across latent classes: the data are assumed to come from `n_classes` subpopulations, each with its own regression of the outcome on the predictors, and the model estimates the regressions, the class sizes and every row's posterior class membership at once. This is clusterwise, latent class or mixture regression (DeSarbo and Cron 1988; Wedel and DeSarbo 1995).

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
)

```

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", "binomial", "poisson", "negative_binomial" or "ordinal" (proportional-odds cumulative logit; see <i>Families</i>).
id	NULL, or the name of the column identifying groups (persons, schools, ...).
class_level	"observation" (each row has its own class) or "group" (all rows of a group share a class; needs id).
n_group_classes	Number of second-level group classes, for class_level = "observation" with id. 1 fits no second level.
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.
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" (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, besides one start built from the residuals of a pooled regression. Every start runs 50 EM iterations; the better half (at least two) continue to convergence.
max_iter	Maximum EM iterations per start.
tol	Relative convergence tolerance on the log likelihood.
min_variance	Floor for a Gaussian residual variance, relative to the outcome's variance. A class that reaches it is fitting too few rows exactly; such starts are set aside.
seed	NULL or an integer seed. The caller's random state is restored afterwards.
missing	"error" refuses rows with missing values in any variable the model uses; "omit" drops them with a <code>latents_rows_dropped</code> warning stating how many.
select_start	"likelihood" keeps the start with the highest likelihood; "converged" prefers the best start that converged.
vcov_type	Covariance of the estimates stored with the fit: "observed" (inverse observed information), "robust" (sandwich, clustered on the top-level unit, or on <code>id</code> for a single-level fit given <code>id</code>) 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 <code>id</code> , per <code>id</code> group otherwise (constant within it). Pseudo maximum likelihood with the weights scaled to sum to the number of units; <code>vcov_type</code> defaults to "robust" and refuses "observed" and "opg".
random	NULL (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 $\sim 1 + \text{time}$ (needed for anything names cannot say, such as $\sim 0 + \text{time}$, a slope without a random intercept). See the growth mixture section.
random_covariance	How the random-effect covariance differs across classes: "varying" (one per class), "equal" (one shared matrix, as <code>lcmm::hlme()</code> with <code>nwg = FALSE</code>) or "proportional" (a shared matrix times a class-specific scale, the last class's being one; <code>nwg = TRUE</code>).
random_diagonal	TRUE for uncorrelated random effects (a diagonal covariance, as <code>idiag = TRUE</code> in <code>lcmm</code>).
cluster	NULL, or the name of a column grouping the persons (<code>id</code>) into clusters, such as schools, for the multilevel growth mixture model (with <code>random</code>): each cluster belongs to one of <code>n_group_classes</code> group classes, which shifts how probable each trajectory class is for its persons. The clusters are the independent units of the likelihood, the standard errors and the BIC; with <code>n_group_classes = 1</code> the model is the single-level growth mixture with clusters as those units, comparable by BIC with more group classes. See the growth mixture section.

Value

An object of class `latents_mixture_regression`, or `latents_growth_mixture` when random is given. Read it with `as.data.frame()` (the coefficient table) or `get_results()` (every table, by name), and use `predict()`, `plot()`, `summary()`, `coef()`, `vcov()`, `confint()`, `logLik()` and `nobs()` on it. The tables are described on `get_results.latents_mixture_regression()`.

Nesting

`class_level` and `n_group_classes` choose one of three likelihoods.

Single level (`id = NULL`) Every row is its own unit and has its own class.

`class_level = "group"` Every row of a group (`id`) belongs to the same class, so a class is a kind of *group*: the regression mixture for repeated measures, `flexmix(y ~ x | id)`. Each group's evidence is the product of its rows' densities.

`class_level = "observation"` **with** `id` **and** `n_group_classes >= 2` Rows keep their own class, and a second-level group class shifts how probable each regression class is within its groups (Vermunt 2003). With `n_group_classes = 1` the groups only enter the "robust" standard errors, which are then clustered on `id`.

Growth mixture models

With repeated measures of persons (`id`, `class_level = "group"`), a time predictor in formula gives every class its own trajectory: latent class growth analysis (Nagin 2005). `random` adds random effects *within* classes, so persons scatter around their class trajectory: the growth mixture model (Verbeke and Lesaffre 1996; Muthen and Shedden 1999). Within class k , person i 's outcomes are

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

with Z built from `random`. The likelihood is exact (Gaussian, no numerical integration); estimation is EM with the random effects as missing data, finished by quasi-Newton with analytic scores. Such a fit has class `latents_growth_mixture`; its tables, plots and inference are described on `get_results.latents_growth_mixture()` and `plot.latents_growth_mixture()`. Random effects need the gaussian family, `id` and `class_level = "group"`.

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 read with `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 α 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),

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

with ordered thresholds $t_{k1} < \dots < t_{k,C-1}$ per class in place of the intercept, and the parametrization of `MASS::polr()`: a positive slope moves the class towards 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 the classes are identified by how continuous predictors shift the category probabilities: a mixture with no predictors, or with two categories (then the logistic mixture of "binomial"), is refused with `latents_not_identified` unless `class_level = "group"` gives each class assignment several outcomes.

A mixture of Bernoulli regressions with one trial per unit is not identified – any mixture of Bernoullis is again a Bernoulli – so a binary outcome requires `class_level = "group"` (several rows per class assignment) or binomial counts with more than one trial per row. The request is refused otherwise.

Conditions

`latents_bad_argument` for an inconsistent specification; `latents_bad_data` for an outcome outside the family's support or a covariate that varies within a group where it may not; `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 or no predictors; `latents_missing_data` for missing values under `missing = "error"`; `latents_rows_dropped` (warning) under `missing = "omit"`; `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) when a coefficient diverged.

References

- Asparouhov, T., & Muthén, 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.
- Muthén, 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., & Lique, 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")
```

Description

Two-level latent class analysis for categorical indicators: `multilpa()` with discrete indicators in `vars` treated as categorical unless named in `ordinal` or `count` through `...`. Each profile is described by unrestricted response probabilities for categorical items and the corresponding structured parameters for ordinal and count items, and the group classes differ in how probable each profile is for their observations. For models that combine categorical and continuous indicators, call `multilpa()` and name the categorical ones in its `categorical` argument.

Usage

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

Arguments

<code>data</code>	A data frame with one row per observation.
<code>vars</code>	Names of the categorical indicator columns. Numeric, integer, logical, character and factor columns are accepted; factor levels keep their declared order and other types are sorted.
<code>id</code>	Name of the group identifier column, or <code>NULL</code> for a single-level model.
<code>n_profiles</code>	Number of observation-level latent classes (profiles).
<code>n_group_classes</code>	Number of group-level latent classes. Defaults to 2, or to 1 when <code>id = NULL</code> (a single-level model has no second level).
<code>...</code>	Further arguments to <code>multilpa()</code> , such as <code>n_starts</code> , <code>seed</code> , <code>missing</code> , <code>tol</code> or <code>profile_covariates</code> . <code>ordinal</code> and <code>count</code> can name structured discrete indicators; the remaining indicators are categorical. <code>categorical</code> cannot be supplied.

Value

A fitted model of class `multilpa` (or `multilpa_covariates` when membership covariates are given), exactly as `multilpa()` returns it with the corresponding discrete indicator types. Categorical response probabilities are in `get_results(fit, "responses")`, one row per profile, item and category.

Conditions

`latents_bad_argument` when `categorical` is supplied, since its columns are derived from `vars`, `ordinal` and `count` here. Every condition of `multilpa()` can also be raised.

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 <- multilpa(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

Fits individual Gaussian profiles nested within observed groups. A discrete latent group class determines the profile proportions. Profile means and variances are shared across group classes (measurement invariance), and indicators are independent conditional on individual profile membership by default. Full residual covariance and observed-data maximum likelihood for missing indicators are available.

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 = 1e-06,
  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")
)

```

Arguments

data	A data frame containing indicators and a group identifier.
vars	Unique character vector of continuous indicator column names.
id	Name of the observed group identifier column. Character, factor, or numeric identifiers are supported; missing identifiers are not. id has no default: omitting it raises <code>latents_bad_argument</code>, because a forgotten grouping would otherwise be fitted as a different model without saying so. Passing <code>id = NULL</code> explicitly fits a single-level model: the observations are treated as independent, each row is its own unit, and <code>n_group_classes</code> becomes one. That fit raises a <code>latents_single_level</code> message saying so. This is the ordinary Gaussian or latent-class mixture that the two-level model reduces to, and every verb of this package works on it. The unit column is fabricated internally as <code>.observation</code>; it is not returned by <code>get_results(x, "data")</code>, and a data that already has a column of that name raises <code>latents_bad_data</code>. Asking for more than one group class without an <code>id</code> raises <code>latents_bad_argument</code>, because one observation per unit leaves no composition for a second-level class to differ in.
n_profiles	Positive integer number of individual profiles.
n_group_classes	Positive integer number of latent group classes. With <code>id = NULL</code> it is one, and naming anything else is an error.
profile_covariates, group_covariates	Names of numeric columns of data predicting individual-profile and group-class membership through multinomial logits, with the final class as reference. Naming either one fits the one-step covariate model and returns a <code>multilpa_covariates</code> object: profile intercepts differ by group class, profile slopes are shared across group classes unless <code>profile_slopes = "group_class"</code>, and a <code>group_covariate</code> must be constant within each group. Covariates enter in their supplied units, so centre or scale them beforehand if that is what you want. This is one-step maximum likelihood, not a regression on assigned classes; three_step() and r3step() are the staged alternatives. The covariate path supports neither <code>start</code>

	nor fixed, and refuses them by name rather than ignoring them; it supports <code>missing = "fiml"</code> for the indicators, but covariates must be complete. Use one for an ordinary, pooled LPA.
<code>variance_model</code>	Either "varying" (profile-specific variances or covariance matrices) or "equal" (shared across profiles).
<code>n_starts</code>	Positive integer number of EM starts. The first is deterministic: Ward's hierarchical clustering of the standardized indicators (on at most 2000 evenly spaced rows), each cluster described by its own means and spread, which is the start <code>mclust</code> 's name refers to and finds solutions whose profiles differ in shape. The rest are k-means from random centres. When <code>start</code> is supplied, it supplies the first start instead; remaining starts are random initializations. It is ignored when <code>max_iter = 0</code> and <code>start</code> is supplied: that call evaluates the supplied parameters and nothing else, so exactly one start is run whatever <code>n_starts</code> says.
<code>max_iter</code>	Nonnegative integer maximum number of EM updates per start. <code>max_iter = 0</code> performs no update. With <code>start</code> , it returns the model evaluated at exactly those values, whatever <code>n_starts</code> is, so that <code>logLik()</code> scores a parameter set supplied from elsewhere rather than the best of some random initializations that were never asked for. Without <code>start</code> there is nothing to evaluate at, so the random initializations are scored and the highest is returned.
<code>tol</code>	Positive relative log-likelihood tolerance. Convergence requires absolute change no greater than <code>tol * (1 + abs(previous log likelihood))</code> .
<code>min_variance</code>	Positive lower bound on each variance, or each covariance eigenvalue for full covariance, in squared input units. This defines a constrained maximum-likelihood problem. Bound-active estimates are explicitly reported and generate a warning.
<code>seed</code>	Optional random seed: any whole number set. <code>seed()</code> accepts, negative ones included. With a supplied seed, the caller's random-number state is restored on exit.
<code>start</code>	Optional list of means and variances (profiles by indicators), <code>profile_probabilities</code> (group classes by profiles), and <code>group_probabilities</code> (vector). Starting probabilities must be positive. For full covariance, supply covariances (indicators by indicators by profiles); variances may be omitted or must match their diagonals. A categorical model also takes <code>response_probabilities</code> , a list of one profiles-by-categories matrix per indicator named in <code>categorical</code> . That list is read by position unless it is labelled: name its elements after the indicators, or its columns after the categories, and each block is matched to the indicator and category its labels name, so a differently ordered categorical or a differently ordered set of factor levels cannot attach a distribution to the wrong item. A label naming an indicator or a category this fit does not have raises <code>latents_bad_start</code> rather than being aligned by position.
<code>missing</code>	"error" rejects missing indicators; "fiml" maximizes the observed-data likelihood under an ignorable missingness mechanism (MAR). Missing indicators are integrated out, not filled in for likelihood fitting. This holds with membership covariates too; the covariates themselves must be complete (<code>latents_bad_data</code>).
<code>profile_slopes</code>	"shared" (the default) gives each profile covariate one slope per profile logit, common to every group class. "group_class" lets each group class have its

own slopes, so a covariate can predict profile membership differently in different kinds of group (a cross-level interaction); its coefficients are reported with terms such as `z:group_class_1`. It adds $(n_group_classes - 1)$ times as many slope parameters again, each estimated from the groups in its class, so it needs enough groups per class. Naming it without `profile_covariates` raises `latents_bad_argument`.

<code>covariance_model</code>	"diagonal" assumes conditional independence; "full" estimates within-profile residual covariances.
<code>categorical</code>	Character vector naming indicators to treat as categorical. Each is modelled by unrestricted, profile-specific response probabilities over its observed categories, which is the latent class measurement model. Binary, ordinal and unordered indicators are all handled by the same unrestricted 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.
<code>min_probability</code>	Positive lower bound on every categorical response probability, defining a constrained maximum-likelihood problem in the same way <code>min_variance</code> does for Gaussian indicators.
<code>time</code>	Optional name of a column giving each observation's position within its group, such as a wave, occasion or course number. The model does not use it; it is stored so that <code>get_results(x, "sequences")</code> , <code>get_results(x, "sequence_summary")</code> and <code>plot(what = "sequences")</code> can read the assignments back in order. Values must be complete and unique within each group.
<code>fixed</code>	Character vector naming measurement blocks to hold at the values <code>start</code> supplies, instead of estimating them: any of "means", "variances" and "response_probabilities", or "measurement" for every block the model has. Under <code>covariance_model = "full"</code> , "variances" holds the residual covariance matrices. A held block stays exactly as supplied, in every restart, and stops counting towards <code>n_parameters</code> , so this is a different model rather than a different starting point for the same one. <code>start</code> must carry the named blocks; <code>starting_values(fit, what = "measurement")</code> produces them, and <code>fit_staged()</code> wraps the whole two-stage workflow in one call. The mixing parameters cannot be held: they are what a fixed-measurement fit is for.
<code>centering</code>	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 a value reads as a deviation from that unit's average and the profiles become profiles of <i>change</i> rather than of level: this is the within-person, person-mean-centred design (Quintana, 2021; Voelkle, Brose, Schmiedek, & Lindenberger, 2014). "grand" subtracts one mean per indicator, which moves the origin without touching the within-group structure. The offsets are kept on the fit, so <code>get_results(x, "data")</code> still returns the columns you supplied and every verb that checks row alignment still checks it. Centring removes exactly the between-unit variation, so "person" refuses with <code>latents_bad_data</code> when it leaves an indicator constant — which is what happens when a unit has one observation of it. With "person" the group classes become types of <i>change pattern</i> , not types of unit.

volume, shape, orientation

The covariance structure, in the three pieces it is made of. Each profile's covariance decomposes as $\Sigma_k = \lambda_k * D_k * A_k * D_k'$: a *volume* $\lambda_k = |\Sigma_k|^{(1/d)}$, an *orientation* D_k of eigenvectors, and a *shape* A_k , diagonal with determinant one. Constraining the three across profiles gives the fourteen models mclust names with three letters, and this package fits all of them.

volume is "equal" or "varying". shape is "equal", "varying" or "spherical", the last making every indicator's spread equal within a profile, which leaves no orientation to constrain. orientation is "axis" (axis-parallel, a diagonal covariance), "equal" (one orientation shared by every profile) or "varying". Each is NULL by default, which follows variance_model and covariance_model, so a call that names none of them fits exactly what it always did.

volume	shape	orientation	model
equal	spherical	—	EII
varying	spherical	—	VII
equal/varying	equal/varying	axis	EEL, VEL, 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); the two models with a shared orientation and a free shape, EVE and VVE, have no closed form and use the minorize-maximize step of Browne and McNicholas (2014). Parameter counts match mclust's own for all fourteen.

Anything other than EEI, VVI, EEE or VVV is maximized across every profile at once, so it cannot be combined with a held variances block. [parameter_inference\(\)](#) reports Wald standard errors for all fourteen: a constrained structure is differentiated in its own free coordinates (log volumes, determinant-one log shapes and orientations), which have exactly as many dimensions as the structure has parameters, and carried to the reported variances and covariances by the delta method. method = "bootstrap" is available for all fourteen as well.

model The covariance structure named by its mclust code, one of "EII", "VII", "EEI", "VEI", "EVI", "VVI", "EEE", "VEE", "EVE", "VVE", "EEV", "VEV", "EVV", "VVV": the first letter is the volume, the second the shape and the third the orientation, each equal (E) or varying (V) across profiles, with I for axis-aligned (no covariances). The same codes [enumerate_classes\(\)](#) takes. Use it instead of variance_model/covariance_model or volume/shape/orientation, not together with them.

select_start Which start the fit reports. "likelihood", the default, takes the highest log likelihood across starts, preferring a converged start only among those tied with it to within a relative $1e-10$. "converged" takes the highest log likelihood among the converged starts whenever at least one converged, and falls back to every start when none did. Use it when an unconverged start edges out a converged one by a negligible amount, which happens when a membership logit creeps along a flat ridge of the likelihood: the converged start is then a maximum that [parameter_inference\(\)](#) can work with, and the other is not. [summary\(\)](#) and [get_results\(fit, "starts"\)](#) show every start either way.

prior	NULL, the default, for maximum likelihood, or <code>prior_control()</code> for maximum a posteriori estimation of the Gaussian means and covariances under the conjugate prior of Fraley and Raftery (2007) — <code>mclust</code>'s <code>priorControl()</code>. The hyperparameters default to <code>mclust</code>'s <code>defaultPrior()</code> computed from the indicators being fitted, and each M-step is <code>mclust</code>'s own, so a fit from the same start reproduces <code>mclust::Mclust(..., prior = priorControl())</code>. It is defined for EII, VII, EEI, VEI, EVI, VVI, EEE, EEV, VEV and VVV (<code>mclust</code> has none for VEE, EVE, VVE and EVV), for complete continuous indicators without fixed or membership covariates; anything else raises <code>latents_unsupported_prior</code>. The mixing proportions are not given a prior. Following <code>mclust</code>, the fit's <code>log_likelihood</code> is the <i>unpenalized</i> log likelihood evaluated at the posterior mode, and <code>aic</code>, <code>bic</code> and their variants are computed from it with the usual parameter count; EM convergence is judged on its relative change. A prior's MAP step can lower that likelihood, so the decrease check made under maximum likelihood is not applied. <code>parameter_inference(method = "wald")</code> is refused for such a fit (<code>latents_unsupported_inference</code>), because the likelihood's score is not zero at a posterior mode; <code>method = "bootstrap"</code> refits every resample with the same prior. The fit records the request as <code>prior</code> and the resolved hyperparameters, in the indicators' units, as <code>prior_parameters</code>.
noise	FALSE, the default, or TRUE to add a noise component: one more mixture component with a constant density $1 / V$ over the hypervolume V of the data, which absorbs observations no Gaussian profile explains — outliers and scatter — instead of letting them distort a profile (Banfield & Raftery, 1993; <code>mclust</code>'s <code>initialization = list(noise =)</code> and <code>Vinv</code>). V is <code>mclust</code>'s <code>hypvol()</code>: 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 <code>mclust</code> counts it, the model has two more parameters (the proportion and the hypervolume), so its criteria equal <code>mclust</code>'s. A random start puts a tenth of the mass on the noise; a <code>start</code> supplies <code>noise_probability</code> beside <code>profile_probabilities</code>, the two summing to one. Available for continuous, complete indicators with one group class (<code>id = NULL</code>, or <code>n_group_classes = 1</code>) and any of the fourteen covariance structures, alone or with prior; otherwise <code>latents_unsupported_noise</code>. In the fit, the noise component is profile 0, as in <code>mclust</code>'s classification: <code>subject_profiles</code> is 0 for an observation assigned to it, <code>get_results(fit, "assignments")</code> adds a <code>posterior_noise</code> column, <code>get_results(fit, "posteriors")</code> and <code>get_results(fit, "profile_probabilities")</code> have profile-0 rows, and the classification diagnostics count it as a class. <code>means</code>, <code>variances</code>, <code>covariances</code>, <code>profile_probabilities</code> and <code>subject_posteriors</code> describe the Gaussian profiles only, so the latter two sum to one minus the noise share; the fit adds <code>noise_probability</code>, <code>noise_posteriors</code>, <code>n_noise</code> and <code>hypervolume</code>. Verbs that do not yet account for the component — <code>parameter_inference()</code>, <code>three_step()</code>, <code>r3step()</code>, <code>bootstrap_lrt()</code>, <code>starting_values()</code>, bivariate residuals, <code>fit_staged()</code> and the posterior plots — refuse such a fit with <code>latents_unsupported_noise</code>.
acceleration	"squarem", the default, accelerates EM with SQUAREM (Varadhan & Roland, 2008): each cycle extrapolates along the last two EM steps and finishes with an ordinary EM step, and falls back to plain EM whenever the extrapolated point would lower the likelihood, so the iteration stays monotone and converges to

the same kind of maximum. The convergence test is read over a whole cycle, so it is never looser than plain EM's. "none" runs plain EM, which follows the same path as other EM implementations from the same start (mclust, Mplus) and is the choice for step-by-step reproduction. iterations and max_iter count EM steps either way. 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 rather than solving it (a generalized EM step), under either setting; this keeps every iteration monotone and makes the EM map a fixed function, which SQUAREM requires.

family

The model family. "profiles", the default, is the model described above: individual profiles whose prevalences differ across group classes. The other three are group-class-only families with 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 and seed; any other argument is refused with latents_bad_argument. 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. Read it with `get_results.multilpa_additive()`; `summary()`, `plot()`, `coef()`, `vcov()`, `confint()` and `parameter_inference()` work on it. These families are experimental: enumeration over families, bootstrap inference, missing data and covariates are not yet 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 and seed. 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.multilpa_cross_level()`.

`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; "varying" is refused there. 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, as Mplus does, 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. Standard errors are the sandwich: `parameter_inference()` defaults to `vcov_type = "robust"` and refuses "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 refuse a weighted fit. Works with every family, membership covariates, categorical indicators and missing = "fiml".

`ordinal`

Names of indicators in vars to model as ordinal: an (ordered) factor, or whole numbers, 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, $\log P(k | c) / P(k - 1 | c) = a_k - a_{(k-1)} + \eta_{c,k}$, 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. Read the category probabilities and locations with `get_results(fit, "ordinal")`.

`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`); read the means with `get_results(fit, "count_means")`. Ordinal and count indicators take missing = "fiml", weights, two-level fits, membership covariates, Wald and bootstrap inference, `bootstrap_lrt()`, `predict()` and `lta()`; start, fixed, prior, noise and the group-class families do not take them yet (`latents_unsupported_indicator`).

`count_model`

"poisson" (the default) or "negative_binomial": NB2, with mean μ and variance $\mu + \alpha \mu^2$, for counts more variable 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 refused there).

`count_dispersion`

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

Details

Infinite and constant observed indicators are rejected. Each indicator must have at least two distinct observed values. No rows are silently dropped. In FIML mode, fully missing individuals contribute no direct measurement likelihood but receive posterior probabilities from their group's information. Observation counts retain these individuals; the individual-level BIC excludes them through `n_informative`, while the group-level BIC uses all observed groups. Missing patterns can prevent parameter identification; no general identification guarantee is made. Initialization alone uses indicator-mean filling. 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 multiple individuals; these checks are necessary but do not establish identification. By default the highest finite likelihood across starts is returned, even if that start did not converge; `select_start = "converged"` prefers a converged start, and `converged` and `starts` show which was chosen. Profile and group-class labels are arbitrary.

Value

An `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 retain first-occurrence order. `bic` and `bic_groups` use the observed group count; `bic_individual` uses `n_informative`, the number of rows carrying at least one observed indicator. These are alternative conventions, not interchangeable criteria. `boundary` identifies variance bounds; `small_classes` flags effective memberships below one. The fit itself carries no standard errors: `parameter_inference()` computes them from the fit and the data it was fitted to, and `bootstrap_lrt()` tests nested models. No guarantee of global optimality is given, whatever `n_starts` is used. Read the tidy form with `as.data.frame()`; what selects which table.

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

Tidy inference for a covariate fit

Description

Standard errors, tests and intervals for every free parameter of `mutilpa(profile_covariates = )`, at all three levels at once: the measurement model, the profile logits and the group-class logits. Without them a membership coefficient cannot be reported, because nothing distinguishes a real effect from separation.

Computes ordinary maximum-likelihood standard errors from the inverse observed Hessian and delta-method Wald confidence intervals. These are local, asymptotic estimates conditional on the selected numbers of classes. They are not robust sandwich errors and do not account for model selection. Mixture boundary solutions and unidentified Hessians do not admit this calculation.

Usage

```
## S3 method for class 'mutilpa_additive'
parameter_inference(
  x,
  data = NULL,
  level = 0.95,
  step = 1e-04,
  vcov_type = c("observed", "robust", "opg"),
  adjust = .mutilpa_p_adjust_methods,
  method = c("wald", "bootstrap"),
  ...
)

## S3 method for class 'mutilpa_covariates'
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")
)

## S3 method for class 'multilpa_cross_level'
parameter_inference(x, ...)

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")
)

## S3 method for class 'multilpa'
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")
)

## S3 method for class 'multilpa_lta'
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 multilpa fit with inactive variance bounds.
data	Optional. The data frame the model was fitted to; when omitted it is rebuilt from the indicators, identifiers and occasions the fit stores, which round-trip exactly. Supplying it is the stronger check that the caller still holds that frame. A supplied frame is checked exactly against the stored indicators and identifiers; for an older fit without stored indicators, the likelihood provides a weaker consistency check.
level	Confidence level strictly between zero and one.
step	Positive finite-difference step for the analytic score derivative.
vcov_type	"observed" inverts the observed information. "robust" returns the Huber-White sandwich $A^{-1} B A^{-1}$, where B accumulates the outer product of the per-group scores. Groups are the independent units, so robust errors relax the assumption that the Gaussian within-group model is correctly specified. They do not relax the assumption that groups are independent. "opg" inverts the outer product of the per-group scores (the BHHH or outer-product-of-gradients estimate), which estimates the information by the variance of the score rather than by the curvature of the likelihood. It is the estimator glca reports, so it reproduces that package's standard errors; the two agree with "observed" in large samples and can differ noticeably in small ones, where "observed" is usually the more reliable. "robust" and "opg" need more groups than parameters.
adjust	Multiplicity correction applied to p_value to produce p_adjusted, one of the methods <code>stats::p.adjust()</code> accepts. The default "none" leaves the two columns equal: a correction changes what a p-value means, so it is applied only when it is asked for, and the method that was applied is recorded in the adjust attribute. The correction is taken over the tests the table actually reports; bounded parameters carry no test and do not count towards the family.
method	How uncertainty is measured. "wald", the default, inverts the observed information in the estimation coordinates, and 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 the volume, the shape or the orientation across profiles are estimated in log volumes, log shapes whose

determinant-one constraint is written 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 of the structure. The reported variances and covariances are then carried from those coordinates by the delta method, and their natural-scale covariance is singular, as it is for the class probabilities. "bootstrap" resamples *groups* with replacement, refits inside the same covariance family, undoes the relabelling each refit comes back with, and reports the percentile interval and the standard deviation of the replicates. It is available for all fourteen structures. Groups are the resampling unit rather than rows, so the interval carries the same independence assumption as `vcov_type = "robust"` and not the stronger one "observed" makes.

...	Unused by most methods; accepted so every method shares the generic's signature.
iter	Number of resamples when <code>method = "bootstrap"</code> , at least two. The default 199 is enough for a standard error; a 95% percentile interval is steadier at 999 and above.
n_starts, max_iter, tol	Passed to each bootstrap refit. Each replicate is a fresh fit, so <code>n_starts</code> buys the same protection against a local maximum here as it does in <code>multilpa()</code> , at the same multiple of the cost.
seed	Optional seed for the resampling: any whole number <code>set.seed()</code> accepts; a fraction raises <code>latents_bad_argument</code> . The stream is restored on exit, so a seeded call leaves the caller's random state exactly as it found it.
boundary	What to do when categorical response probabilities sit on their lower bound, <code>min_probability</code> . There the likelihood is flat in the logit and the information is singular, so "error", the default, refuses with <code>latents_boundary_fit</code> . "fix" holds those probabilities at the bound and reports every other parameter conditionally on them, the convention Mplus follows; the held probabilities are left out of the table (or, for a covariate fit, reported with NA standard errors) and named in the result's <code>fixed_at_bound</code> attribute.

Details

Class-membership coefficients are on the multinomial logit scale with the final profile and the final group class as references, so a coefficient is a log odds against that reference. The tests are Wald tests and inherit the usual caveat: they are unreliable for a coefficient driven to the boundary by separation, which the size of the estimate and its standard error together will reveal.

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` (and previous for a second-order fit). `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 holds nothing at a bound: with no bound active `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 free parameter and the same columns, in the same order, that `parameter_inference()` returns for every other fitted class: level ("measurement", "profile" or "group"), outcome (which profile or group class the coefficient predicts, or which profile a measurement parameter belongs to), term, parameter ("mean", "variance", "covariance", "response" or "coefficient"), estimate, standard_error, statistic, p_value, p_adjusted, conf_low and conf_high. Variances and covariances are reported in their natural units, with standard errors carried through the delta method from the log and log-Cholesky coordinates on which they are estimated. Categorical indicators contribute one "response" row per profile, indicator and category, named indicator:category: the probability of that category, with its standard error carried from the multinomial logits it is estimated on, and no Wald test, as for the covariate-free model.

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 their own units, probabilities as probabilities. A parameter whose null value is on the boundary of its space – a variance, a probability, a residual variance on the covariance diagonal – carries NA for statistic, p_value and p_adjusted, because a Wald test against that boundary is not a question worth asking; the interval still is. 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) on the log scale, then mapped back with the same standard error, so they stay inside the parameter's range and widen towards the far side of a bound; every other interval is estimate $\pm z \times \text{standard_error}$. `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 is a constant of this likelihood, so it contributes no row here and no row or column to the information matrix. The table is on the natural scale, where every class probability is reported, so it has one row per estimated natural coefficient: `n_parameters` rows plus one for each set of probabilities whose reference category the estimation scale drops. The held values still enter the likelihood at the values they were held at, so every standard error, interval and p-value is conditional on that measurement solution and does not propagate its uncertainty. See `fit_staged()` for what that conditioning means. For a `fit_staged()` result, `method = "bootstrap"` does propagate it: each resample refits both stages, and the table has a row for every parameter, the measurement included (attribute `stages_resampled` is 2).

With `method = "bootstrap"` the table has the same columns and the same rows, estimate is still the fitted value, standard_error is the standard deviation of the replicates and conf_low/conf_high are the percentile interval. statistic, p_value and p_adjusted are NA throughout: putting a normal approximation back on top of the replicates the interval was read from is the assumption this path exists to avoid, so the interval is the inference. Its attributes are covariance (the covariance of the replicates, which `vcov()` returns), 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, because the estimation-scale chart is what this path does without.

Diagnostics of the fit as a whole travel as attributes rather than as columns repeated down every row: covariance and covariance_unconstrained (natural and estimation-scale covariance of the estimates), hessian, gradient, scaled_score, condition_ratio, level, step, vcov_type, fixed (the measurement blocks this fit held, `character()` when none) and adjust. When `vcov_type` is "robust", `group_scores` holds the groups-by-parameters score matrix and `scaling_correction`

the MLR scaling correction factor $\text{tr}(A^{-1} B) / q$, used for scaled likelihood-ratio difference tests.

Conditions

latents_no_converge for an unconverged fit, latents_boundary_fit when a variance, a response probability or a mixing probability sits 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, and latents_too_few_groups for vcov_type = "robust" with fewer independent groups than reported parameters. A fit whose score is still far from zero is reported with a latents_unconverged warning rather than refused. A fit made with multilpa(prior =) is refused by the Wald path with latents_unsupported_inference (it sits at a posterior mode, where the likelihood's score is not zero), and a fit with a noise component by either path with latents_unsupported_noise.

method = "bootstrap" adds latents_unsupported_inference for a fit that holds a measurement block through multilpa(fixed =) — the held values came from another fit and resampling these data does not resample them; a `fit_staged()` result is bootstrapped by staging again instead — and latents_bootstrap_failed when fewer than two resamples produced a usable fit, whose message carries the first reason one gave. Resamples that fail or do not converge are dropped with a latents_bootstrap_dropped warning naming how many, rather than being silently left out of the count.

Examples

```
# Two separated profiles, 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
)
example_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)
)
fit <- multilpa(example_data, c("y1", "y2"), "school", n_profiles = 2,
  n_group_classes = 2, profile_covariates = "x",
  n_starts = 2, seed = 1)
parameter_inference(fit, example_data)
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)

# Many tests in one table: name the correction, do not apply one by stealth.
```

```
parameter_inference(fit, adjust = "BH")

# A structure that constrains the shape across profiles (VEI) is charted in
# its own free coordinates, so it has Wald standard errors too.
shaped <- mutilpa(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)

# The bootstrap resamples schools and refits inside the same family.
parameter_inference(shaped, method = "bootstrap", iter = 10, n_starts = 1,
                  seed = 1)
```

```
parameter_inference.mutilpa_transitions
```

Wald inference for a latent transition model

Description

Standard errors, tests and intervals for every parameter of an `lta()` fit: the measurement model, the group-class shares, each group class's initial profile distribution and its transition matrix. The scores follow the Fisher identity (each is an expected count from the forward-backward pass minus the count the fitted probabilities imply), and the observed information is the numerical Jacobian of that analytic score.

Usage

```
## S3 method for class 'mutilpa_transitions'
parameter_inference(
  x,
  data = NULL,
  level = 0.95,
  step = 1e-04,
  vcov_type = c("observed", "robust", "opg"),
  adjust = .mutilpa_p_adjust_methods,
  method = c("wald", "bootstrap"),
  iter = 199L,
  n_starts = 10L,
  max_iter = 1000L,
  tol = 1e-08,
  seed = NULL,
  boundary = c("error", "fix")
)

## S3 method for class 'mutilpa_transitions'
vcov(
  object,
  data = NULL,
```

```

    step = 1e-04,
    vcov_type = c("observed", "robust", "opg"),
    scale = c("natural", "unconstrained"),
    ...
)

## S3 method for class 'multilpa_transitions'
confint(object, parm, level = 0.95, data = NULL, ...)

```

Arguments

x	A fitted multilpa_transitions model from <code>lta()</code> .
data	Optional. The data frame the model was fitted to. The fit stores its indicators, codes, groups and occasions, so it is not needed; when given it must reproduce them.
level	Confidence level of the intervals.
step	Finite-difference step for the observed information.
vcov_type	"observed" (observed information), "robust" (the sandwich clustered on groups, which are the sequences) or "opg" (the outer product of the group scores).
adjust	Multiplicity correction for p_adjusted, one of the methods <code>stats::p.adjust()</code> accepts.
method	"wald" (the default) or "bootstrap". The bootstrap resamples persons with replacement, refits the same specification, and matches each replicate's profiles and group classes to the original's; standard_error is the replicates' standard deviation, conf_low and conf_high their percentile interval, and statistic, p_value and p_adjusted are NA. It needs data.
iter, n_starts, max_iter, tol, seed	Bootstrap resamples, the controls of each refit, and an optional seed (the caller's random state is restored).
boundary	Accepted for the generic. Transition inference holds nothing at a bound: with no bound active "fix" gives the same table as "error", and a fit with an active bound raises latents_boundary_fit under either.
object	A fitted multilpa_transitions model.
scale	"natural" (probabilities) or "unconstrained" (logits).
...	Unused.
parm	Parameters to report, as names or positions of the natural coefficients; all by default.

Value

A base data.frame with one row per natural parameter and the columns of `parameter_inference()`: level, outcome, term, parameter, estimate, standard_error, statistic, p_value, p_adjusted, conf_low and conf_high, named as `coef()` names them. 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. Probabilities and variances carry no Wald test. Attributes covariance (natural scale), covariance_unconstrained, vcov_type and level.

Conditions

latents_no_converge for an unconverged fit; latents_boundary_fit when a variance, a response, initial or transition probability, or a group-class share sits at its bound, or a transition row was never informed (a profile no group occupies before its last occasion); latents_singular_information when the information cannot be inverted; latents_bad_inference_data when a supplied data does not reproduce the fit; latents_too_few_groups for robust or OPG errors with no more groups than parameters; latents_bad_argument for method = "bootstrap" without data.

Examples

```
fit <- lta(subset(course_engagement, student <= 40),
          c("browse", "lectures", "forum_read"), "student",
          n_profiles = 2, time = "sequence", n_starts = 2, seed = 1)
parameter_inference(fit)
```

```
plot.latents_comparison
```

Plot a model comparison

Description

Each model's AIC, BIC and sample-size adjusted BIC, lowest best; the model with the smallest BIC is marked.

Usage

```
## S3 method for class 'latents_comparison'
plot(x, main = NULL, subtitle = NULL, ...)
```

Arguments

x	A latents_comparison from <code>compare_models()</code> .
main, subtitle	Title and subtitle.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```
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", seed = 1)
comparison <- compare_models(trajectories = trajectories, growth = growth)
if (requireNamespace("ggplot2", quietly = TRUE)) plot(comparison)
```

```
plot.latents_family_enumeration
```

Plot information criteria across group-class families

Description

One line per model code, criterion against the number of group classes; models are told apart by colour, point shape and line type.

Usage

```
## S3 method for class 'latents_family_enumeration'
plot(
  x,
  criterion = c("bic", "aic", "bic_individual"),
  main = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

x	A latents_family_enumeration result.
criterion	"bic", "aic" or "bic_individual".
main, subtitle	Optional title and subtitle.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```

set.seed(1)
ratings <- data.frame(
  team = rep(seq_len(40), each = 6),
  climate = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
    rnorm(240))
grid <- enumerate_classes(ratings, "climate", "team",
  family = c("additive", "dispersion"),
  n_group_classes = 1:2, n_starts = 3, seed = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) plot(grid)

```

plot.latents_growth_mixture

Plot a growth mixture model

Description

Plot a growth mixture model

Usage

```

## S3 method for class 'latents_growth_mixture'
plot(
  x,
  what = c("trajectories", "individuals", "coefficients", "random", "classification",
    "posteriors"),
  time = NULL,
  facet = FALSE,
  persons = 12L,
  max_persons = 120L,
  spread = 0.8,
  level = 0.95,
  main = NULL,
  subtitle = NULL,
  ...
)

```

Arguments

x	A latents_growth_mixture fit.
what	"trajectories" (each class's mean trajectory with its confidence band, the spread of its persons' own trajectories, and the observed class means at each time), "individuals" (a sample of persons across each class's range of classification certainty: observations, their class trajectory dashed, their own predicted curve solid), "coefficients" (estimates with confidence intervals and p-values, by term), "random" (predicted random effects with the class's model-implied 95% ellipse; degenerate covariances are flagged), or "classification"

	(one bar per person, split by their posterior class probabilities). "posteriors" is a synonym of "classification".
time	The numeric variable to draw along; by default the first numeric variable of random.
facet	For "trajectories", TRUE draws one panel per class.
persons	For "individuals", how many persons to show, or a vector of their identifiers.
max_persons	For "trajectories", the most persons drawn behind the class trajectories (an even sample by class); 0 draws none.
spread	For "trajectories", the share of persons' class-implied trajectories the light band holds (from the random effects); 0 hides it.
level	Confidence level of the bands and intervals.
main, subtitle	Title and subtitle; defaults describe the view.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```
few_students <- subset(growth_scores, student <= 60)
fit <- mixture_regression(score ~ wave, few_students, n_classes = 2,
  id = "student", class_level = "group",
  random = "intercept", seed = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) plot(fit)
```

```
plot.latents_mixture_regression
```

Plot a mixture-of-regressions fit

Description

A fit that classifies persons (id with class_level = "group") is a trajectory model, and draws the trajectory views of `plot.latents_growth_mixture()` (ggplot2): "trajectories" (the default), "individuals", "coefficients" and "classification". Any other fit draws base-graphics views: "fitted" (the default), "coefficients" and "posteriors".

Usage

```
## S3 method for class 'latents_mixture_regression'
plot(
  x,
  what = NULL,
  predictor = NULL,
  level = 0.95,
```

```

    main = NULL,
    subtitle = NULL,
    time = NULL,
    facet = FALSE,
    persons = 12L,
    max_persons = 120L,
    ...
  )

```

Arguments

x	A fit from <code>mixture_regression()</code> .
what	For a trajectory model: "trajectories" (each class's mean trajectory with its confidence band and the observed class means), "individuals", "coefficients" (estimates with intervals and p-values) or "classification" ("posteriors" is a synonym). Otherwise: "fitted" (the outcome against one numeric predictor, rows marked by modal class, each class's regression line), "coefficients" or "posteriors" (each unit's largest posterior probability, by class).
predictor	For "fitted", the numeric predictor for the horizontal axis; defaults to the first one in the formula.
level	Confidence level of intervals and bands.
main	Plot title; NULL for the default.
subtitle	For a trajectory model, the subtitle; NULL for the default.
time, facet, persons, max_persons	For a trajectory model, as in <code>plot.latents_growth_mixture()</code> .
...	Unused.

Value

For a trajectory model, a ggplot object (raises `latents_missing_package` without `ggplot2`); otherwise x, invisibly, called for its plot.

Examples

```

fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                        n_starts = 3, seed = 1)

plot(fit)
plot(fit, what = "coefficients")

```

plot.latents_pooled	<i>Plot a pooled multiply imputed fit</i>
---------------------	---

Description

One row per parameter: each imputation's aligned estimate as an open circle and the pooled estimate as a filled circle with its interval, so the between-imputation spread is visible next to the total uncertainty.

Usage

```
## S3 method for class 'latents_pooled'
plot(x, level = c("profile", "group", "measurement"), main = NULL, ...)
```

Arguments

x	A latents_pooled object.
level	Which parameter level to show: "profile" (the profile membership coefficients or probabilities), "group" or "measurement".
main	Plot title; NULL for the default.
...	Unused.

Value

x, invisibly. Called for the side effect of drawing.

Examples

```
set.seed(1)
completed <- lapply(1:2, 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)

plot(pooled)
```

```
plot.latents_transition_enumeration
```

Plot information criteria across transition models

Description

The criterion against the number of profiles, one line per number of group classes (and structure), told apart by colour, shape and line type.

Usage

```
## S3 method for class 'latents_transition_enumeration'
plot(
  x,
  criterion = c("bic", "aic", "bic_individual"),
  main = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

x	A latents_transition_enumeration result.
criterion	"bic", "aic" or "bic_individual".
main, subtitle	Optional title and subtitle.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```
few_students <- subset(course_engagement, student <= 50)
grid <- enumerate_classes(few_students, c("browse", "lectures"),
                          "student", time = "sequence", n_profiles = 2:3,
                          n_group_classes = 1, n_starts = 2, seed = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) plot(grid)
```

plot.mutilpa

Plot a fitted multilevel latent profile model

Description

Draws one view of a fit as a ggplot object, which can be printed, saved with ggplot2::ggsave() or styled further with + ggplot2::theme(). Every view shares one profile order (largest first), one profile share (the posterior share every table reports) and one set of Okabe-Ito colours paired with point shapes, so a profile looks the same in every view. Series are labelled directly rather than through a legend.

Usage

```
## S3 method for class 'mutilpa'
plot(
  x,
  what = c("profiles", "bars", "heatmap", "raincloud", "parallel", "pairs", "responses",
           "probabilities", "sequences", "sizes", "entropy", "posteriors", "avepp", "all"),
  data = NULL,
  scale = c("raw", "standardized"),
  category = "last",
  labels = TRUE,
  intervals = TRUE,
  cell_labels = TRUE,
  main = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

x	A fitted mutilpa model.
what	Which view to draw. <code>plot_views()</code> lists every value with its group and a one-line description. The measurement model: "profiles" draws one line per profile across the continuous indicators. "bars" draws the same means as grouped bars that start at zero. "heatmap" draws them as a diverging grid of observed standard deviations from each indicator's observed mean, the quickest read when there are many indicators or profiles; for a fit whose indicators are all categorical it draws the response probabilities instead. "raincloud" shows what the means summarize: for the cases assigned to each profile, a density, the quartiles and the observations of each indicator. "parallel" draws every case as a line across the indicators, one panel per profile, faded by the certainty of its assignment. "pairs" draws a scatter-plot matrix of the indicators with each profile's 95% ellipse from the fitted covariances, the view that shows the covariance structure. "responses" is the categorical counterpart of "profiles": one line per profile, showing the probability of a chosen category. The two-level structure: "probabilities" plots profile prevalence within each group class. "sequences" draws one row per group and one column per position, filled with the assigned profile; it needs a fit made with <code>time =</code> . Classification: "sizes" draws each profile's effective count, the posterior mass it carries. "posteriors" draws the posterior probability of each case's assigned profile and "entropy" each case's entropy relative to a flat posterior, one strip per profile with its mean marked. A one-profile fit refuses both with an error of class <code>latents_nothing_to_plot</code> . "avepp" draws the average posterior probability matrix: rows are assigned profiles, columns profiles, and the diagonal is the avePP usually reported. "all" returns every view the fit has the ingredients for.
data	Optional. The data frame the model was fitted to. A fit carries the columns it was built from, so intervals are drawn without it; pass it only to compute them from a different frame.
scale	For the mean views and "raincloud", "raw" keeps each indicator in its input units and "standardized" divides its deviation from the observed mean by its observed standard deviation, the same map as <code>as.data.frame(scale = "standardized")</code> .
category	For <code>what = "responses"</code> , which category's probability to plot: "last", "first", or a single category label or index.
labels	TRUE labels each series at its right end; FALSE uses a legend instead.
intervals	For "profiles" and "responses", TRUE draws 95% intervals when the fit has standard errors (a probability's is clipped to <code>[0, 1]</code>). A fit without them is drawn without whiskers and its subtitle says why.
cell_labels	For <code>what = "sequences"</code> , TRUE prints the profile number in each cell while the grid is sparse enough to hold one, so the profile is not carried by colour alone.
main, subtitle	Title and subtitle. NULL uses the view's own.
...	Nothing further is accepted; an unknown argument raises an error of class <code>latents_bad_argument</code> . Style the returned plot with <code>ggplot2</code> .

Details

The plots need the `ggplot2` package, which `latents` suggests rather than requires; without it a plot is refused with an error of class `latents_missing_package`.

"profiles" and "bars" draw 95% Wald intervals when the fit has standard errors. Profile numbers are arbitrary, so two fits must have their labels aligned before their plots are compared. With `scale = "standardized"` the standard deviations are the observed indicator standard deviations, not the within-profile ones, so the values are comparable across indicators but are not effect sizes.

Value

A `ggplot` object; for `what = "all"`, a `latents_plots` list of them, named by view, that draws every one when printed.

Errors

`latents_missing_package` without `ggplot2`; `latents_no_continuous`, `latents_no_categorical`, `latents_no_time` and `latents_nothing_to_plot` when the fit lacks what a view needs; `latents_bad_argument` for an argument the method does not use.

References

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

See Also

[plot_views\(\)](#) for the catalogue of views.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  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)
  plot(fit)
  plot(fit, scale = "standardized")
  plot(fit, what = "bars")
  plot(fit, what = "pairs")
  plot(fit, what = "avepp")
}
```

plot.mutilpa_additive

Plot an additive group-class fit

Description

"means" shows each group class's mean per indicator with its Wald interval (when the fit supports inference). "variances" sets the shared within-group variance beside each between-group variance. "intercepts" shows every group's posterior intercept mean, marked by its modal class, with the class means over them. Classes are distinguished by colour and by point shape.

Usage

```
## S3 method for class 'mutilpa_additive'
plot(
  x,
  what = c("means", "variances", "intercepts"),
  level = 0.95,
  main = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

x	A mutilpa_additive fit.
what	Which view.
level	Confidence level of the intervals.
main, subtitle	Optional title and subtitle.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(1)
  ratings <- data.frame(
    team = rep(seq_len(40), each = 6),
    climate = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
      rnorm(240),
    support = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
      rnorm(240))
  fit <- mutilpa(ratings, c("climate", "support"), "team",
    n_group_classes = 2, family = "additive", n_starts = 3,
```

```

                                seed = 1)
  plot(fit)
  plot(fit, what = "intercepts")
}

```

plot.multilpa_bootstrap_lrt

Plot a simulated bootstrap null distribution

Description

Draws the replicate likelihood-ratio statistics as a histogram with the observed statistic marked, so that the p-value can be read as a tail area of the distribution that was actually simulated. Invalid replicates carry no statistic and are counted in the subtitle rather than dropped silently.

Usage

```

## S3 method for class 'multilpa_bootstrap_lrt'
plot(
  x,
  main = NULL,
  subtitle = NULL,
  palette = NULL,
  style = .multilpa_style(),
  ...
)

```

Arguments

x	An multilpa_bootstrap_lrt result.
main, subtitle	Panel title and secondary line, or NULL for defaults.
palette	A vector of at least two colours: the histogram fill and the observed-statistic marker. NULL uses the package palette.
style	A list of visual constants, as built by .multilpa_style().
...	Further named visual constants, merged into style.

Value

The input, invisibly. Called for the side effect of drawing.

Conditions

latents_nothing_to_plot when no replicate produced a finite statistic.

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 <- mutilpa(example_data, "score", "school", n_profiles = 1,
  n_group_classes = 1, n_starts = 2, seed = 1)
larger <- mutilpa(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)
plot(comparison)
```

plot.mutilpa_covariates

Plot a covariate model

Description

Draws the parts of a covariate fit that are still fixed quantities: the measurement model, the assignments and the classification diagnostics, as `plot.mutilpa()` draws them. Profile prevalence cannot be drawn: a covariate model has no single prevalence vector, because prevalence varies with each unit's covariates, so asking for it is refused rather than answered with an average that no unit has.

Usage

```
## S3 method for class 'mutilpa_covariates'
plot(
  x,
  what = c("profiles", "bars", "heatmap", "raincloud", "parallel", "pairs", "responses",
    "sequences", "sizes", "entropy", "posteriors", "avepp", "all"),
  data = NULL,
  scale = c("raw", "standardized"),
  category = "last",
  labels = TRUE,
  intervals = TRUE,
  cell_labels = TRUE,
  main = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

`x` A fitted mutilpa_covariates model.

`what` The view; every view of `plot.mutilpa()` except "probabilities".

`data, scale, category, labels, intervals, cell_labels, main, subtitle, ...`
As in `plot.mutilpa()`.

Value

A ggplot object; for `what = "all"`, a `latents_plots` list.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  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
  )
  example_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)
  )
  fit <- mutilpa(example_data, c("y1", "y2"), "school", n_profiles = 2,
    n_group_classes = 2, profile_covariates = "x",
    n_starts = 2)
  plot(fit)
}
```

plot.mutilpa_cross_level

Plot a cross-level fit

Description

"profiles" shows the individual profiles' means per indicator; "group_means" the group classes' means of the group means; "composition" the profile shares within each group class. Profiles and classes are told apart by colour and point shape.

Usage

```
## S3 method for class 'mutilpa_cross_level'
plot(
  x,
  what = c("profiles", "group_means", "composition"),
  main = NULL,
```

```

    subtitle = NULL,
    ...
  )

```

Arguments

x	A mutilpa_cross_level fit.
what	Which view.
main, subtitle	Optional title and subtitle.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```

set.seed(1)
ratings <- data.frame(
  team = rep(seq_len(40), each = 6),
  climate = rep(rnorm(40, rep(c(-1, 1), each = 20), 0.5), each = 6) +
    rnorm(240, sample(c(-1, 1), 240, replace = TRUE), 0.6))
fit <- mutilpa(ratings, "climate", "team", n_profiles = 2,
  n_group_classes = 2, family = "full_cross_level",
  n_starts = 3, seed = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) plot(fit)

```

plot.mutilpa_enumeration

Plot a class-enumeration grid

Description

what = "enumeration" draws information criteria against the number of profiles, one panel per criterion and one line per covariance model and number of group classes. A candidate that failed to converge is marked with a cross on its panel's floor, so a gap in a line reads as a failure and not as a missing candidate. what = "tree" draws how profiles split as more are added: each row is one solution divided into its profiles by posterior share, and bands carry the posterior mass shared by a profile and one in the next solution, coloured by lineage. A band that forks is a profile split in two; bands that merge or cross are cases reshuffled.

Usage

```

## S3 method for class 'mutilpa_enumeration'
plot(
  x,
  what = c("enumeration", "tree"),

```

```

criterion = c("aic", "bic_groups", "bic_individual", "icl_individual"),
combine = TRUE,
labels = TRUE,
mark_minimum = TRUE,
main = NULL,
subtitle = NULL,
...
)

```

Arguments

x	A mutilpa_enumeration result from enumerate_classes() , enumerate_lpa() or enumerate_lca() .
what	"enumeration" (the default) or "tree".
criterion	For "enumeration", one or more criterion columns of <code>as.data.frame(x)</code> , such as "bic_individual" or "sabic_groups". The default draws AIC, BIC under both sample-size conventions and ICL counted over individuals; a criterion identical at both levels, as in a single-level grid, is drawn once.
combine	TRUE draws several criteria as panels of one plot; FALSE returns one plot per criterion.
labels	TRUE labels each series at its right end.
mark_minimum	TRUE rings each criterion's lowest value among converged candidates. This marks an extremum; it does not select a model.
main, subtitle	Title and subtitle. NULL uses the view's own.
...	Nothing further is accepted; an unknown argument raises an error of class <code>latents_bad_argument</code> .

Value

A ggplot object, or with `combine = FALSE` and several criteria a `latents_plots` list of them, one per criterion.

Errors

`latents_unknown_criterion` for a criterion that is not a column of the grid; `latents_nothing_to_plot` when no converged candidate has a finite value, or when a tree has fewer than two numbers of profiles for every model.

References

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

Examples

```

if (requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(7)
  example_data <- data.frame(score_a = rnorm(120), score_b = rnorm(120))
  candidates <- enumerate_lpa(example_data, c("score_a", "score_b"),

```

```

                                n_profiles = 1:3, n_starts = 2)
  plot(candidates)
  plot(candidates, what = "tree")
}

```

plot.mutilpa_lta	<i>Plot transition probabilities of a general transition fit</i>
------------------	--

Description

One panel per group class (and occasion, when transitions vary): a tile per origin and destination, labelled with its probability.

Usage

```

## S3 method for class 'mutilpa_lta'
plot(x, main = NULL, subtitle = NULL, ...)

```

Arguments

x	A mutilpa_lta fit.
main, subtitle	Optional title and subtitle.
...	Unused.

Value

A ggplot object. Raises latents_missing_package without ggplot2.

Examples

```

fit <- lta(course_engagement, c("browse", "lectures"), "student",
         n_profiles = 2, time = "sequence",
         transition_covariates = "previous_grade", n_starts = 2, seed = 1)
if (requireNamespace("ggplot2", quietly = TRUE)) plot(fit)

```

plot.mutilpa_transitions	<i>Plot a fitted latent transition model</i>
--------------------------	--

Description

The measurement model is the one `mutilpa()` fits, so every measurement and classification view it draws is available here. `what = "transitions"` is the view this family adds: the estimated transition matrix, one panel per group class, with a row that has no data support labelled in parentheses because it is uniform by construction rather than estimated.

Usage

```
## S3 method for class 'mutilpa_transitions'
plot(
  x,
  what = c("transitions", "profiles", "bars", "heatmap", "responses", "sequences",
    "sizes", "entropy", "posteriors", "avepp", "all"),
  data = NULL,
  scale = c("raw", "standardized"),
  category = "last",
  labels = TRUE,
  cell_labels = TRUE,
  main = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

x	A fitted mutilpa_transitions model.
what	The view to draw. "transitions" draws the estimated transition matrix, one panel per group class. "profiles", "bars" and "heatmap" draw the measurement model, "responses" the categorical response curves, "sequences" each group's profile at each occasion, and "sizes", "entropy", "posteriors" and "avepp" the classification diagnostics. "all" returns every view this fit has the ingredients for.
data	Optional. Accepted for consistency with plot.mutilpa() ; this family's views draw point estimates, and parameter_inference() reports the standard errors.
scale	"raw" keeps the indicators in their own units; "standardized" divides by each indicator's observed standard deviation.
category	For "responses", which category to draw.
labels	Whether to label series directly.
cell_labels	For "sequences", whether to print the profile number in each cell.
main, subtitle	Title and subtitle. NULL uses the view's own.
...	Nothing further is accepted; an unknown argument raises an error of class <code>latents_bad_argument</code> .

Value

A ggplot object; for what = "all", a `latents_plots` list.

See Also

[get_tna\(\)](#) and [get_group_tna\(\)](#) to draw the transitions as a network instead, and [plot.mutilpa\(\)](#) for the same views on a cross-sectional fit.

Examples

```

if (requireNamespace("ggplot2", quietly = TRUE)) {
  fit <- lta(subset(course_engagement, student <= 40),
            c("browse", "lectures", "forum_read"), "student",
            n_profiles = 2, time = "sequence", n_starts = 2)
  plot(fit, what = "transitions")
  plot(fit, what = "profiles")
}

```

plot_views

The plots this package can draw

Description

Returns the catalogue of what = values accepted by the plot() methods, with the group each belongs to, so the available views can be listed rather than recalled from a help page.

Usage

```
plot_views()
```

Value

A base data.frame with one row per plot type and the columns type (the value to pass as what), group ("measurement", "structure", "diagnostics", "selection" or "every"), and description.

Examples

```
plot_views()
```

pool_imputations

Fit a model to multiply imputed data and pool it with Rubin's rules

Description

Fits `multilpa()` to every completed data set, aligns each fit's profile and group-class labels to the first one, and pools the estimates with Rubin's rules. This is the route for missing *covariates*, which `missing = "fiml"` cannot integrate out: the model conditions on covariates, so it has no distribution for them to integrate over. Impute them first (for example with the mice package, including the indicators and the group identifier in the imputation model), then pass the completed data here.

Usage

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

Arguments

<code>imputed</code>	The completed data sets: a list of at least two data frames with the same columns and rows in the same order, or a <code>mids</code> object from the <code>mice</code> package (which is then completed here).
<code>vars, id, n_profiles</code>	As in <code>multilpa()</code> .
<code>...</code>	Further arguments for <code>multilpa()</code> , such as <code>n_group_classes</code> , <code>profile_covariates</code> , <code>group_covariates</code> , <code>categorical</code> , <code>n_starts</code> and <code>seed</code> . The same arguments fit every imputation.
<code>level</code>	Confidence level of the pooled intervals.
<code>vcov_type, adjust</code>	As in <code>parameter_inference()</code> , applied to every imputation's fit.

Details

Pooling is on the reported (natural) scale: for each parameter 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, which is what maximum-likelihood inference assumes. Pooled intervals are symmetric t intervals on this scale; unlike the transformed Wald intervals of `parameter_inference()`, they can extend outside the parameter space for probabilities or variances.

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.

Choosing the imputation model

Rubin's rules are only as good as the imputations. The covariate and the indicators are related through the latent classes, so an imputation model that is linear in the indicators (mice's defaults) misses that structure: in `validation/pool-imputations-coverage.R` (a membership slope with 30% of its covariate missing at random given the indicators, 200 replications, ten imputations) 95% intervals covered 0.925 under predictive mean matching and 0.935 under Bayesian linear regression, against 0.995 when the covariate was approximately drawn from its conditional distribution using known generating parameters. That oracle arm omits uncertainty in the imputation parameters and is not a validation of proper multiple imputation or nominal interval coverage. The study is one design, not a general guarantee for any method. Use an imputation model that accounts for the latent structure and clustering, includes the indicators and relevant nonlinearities, and propagates parameter uncertainty. Imputing separately within assigned classes alone ignores classification uncertainty. Complete-case coverage was 0.955 in that design, but selecting on the indicators changes their within-class distributions; the usual logistic case-control slope argument does not guarantee unbiased slopes when the outcome is latent and its measurement model is estimated.

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 an imputation's fit or inference fails (its message names the imputation and carries the original reason), since Rubin's rules need every imputation; `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)
# Two stand-in completed data sets. With real missing covariates, pass the
# `mids` object mice::mice() returns instead.
completed <- lapply(1:2, 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_growth_mixture

Prediction for a growth mixture model (not implemented)

Description

Predictions for new persons are not implemented yet. The fitted class trajectories and each fitted person's own predicted curve are tables of `get_results.latents_growth_mixture()`: "trajectories" and "individual".

Usage

```
## S3 method for class 'latents_growth_mixture'
predict(object, ...)
```

Arguments

object	A latents_growth_mixture fit.
...	Unused.

Value

Never returns; raises latents_unsupported_prediction.

Examples

```
few_students <- subset(growth_scores, student <= 60)
fit <- mixture_regression(score ~ wave, few_students, n_classes = 2,
  id = "student", class_level = "group",
  random = "intercept", seed = 1)
# Prediction is refused; get_results(fit, "trajectories") has the curves
try(predict(fit))
```

predict.latents_mixture_regression

Predict from a mixture-of-regressions fit

Description

Predict from a mixture-of-regressions fit

Usage

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

Arguments

object	A fit from <code>mixture_regression()</code> .
newdata	NULL for the fitted rows, or a data frame with the predictors (and the membership covariates and id the model uses).
type	"response": the mean outcome averaged over the classes with their prior probabilities (the covariates, but not the outcome, inform them). "class_response": the mean outcome under every class, one row per row and class. "posterior": the posterior class probabilities given the outcome, which newdata must then contain. "probabilities" (ordinal family only): the probability of every category under every class. For the ordinal family the "mean outcome" is the expected category score, the categories scored 1 to C in order.
...	Unused.

Value

A base data.frame. For "response": row and fitted. For "class_response": row, class, prior and fitted, one row per data row and class. For "posterior": the same columns as `get_results(fit, "assignments")`. For "probabilities": row, class, category, prior and probability, one row per data row, class and category. Raises `latents_bad_argument` for "probabilities" with another family.

Examples

```
fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                          n_starts = 3, seed = 1)
new_students <- data.frame(hours = c(2, 10))
predict(fit, new_students, type = "class_response")
```

predict.mutilpa

Classify new observations with a fitted model

Description

Evaluates the fitted mixture on new rows: the E-step of the fit, at its estimates, applied to data it has not seen. Rows go through the fit's own preparation – the same categorical levels, the same centering, the same missing-data handling – so a prediction on the training data reproduces the fit's own posteriors exactly.

Usage

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

Arguments

object	A fit from <code>mutilpa()</code> or <code>mutilca()</code> without membership covariates.
newdata	NULL for the fitted rows, or a data frame with the fit's indicator columns (and its id column for a two-level fit).
type	"class": the modal profile and its probability, and the modal group class. "posterior": the same with every profile's probability. "density": each row's log density under the fitted mixture, marginal over the group classes (the density of a row from a new group), for scoring, outlier detection or held-out likelihood.
...	Unused.

Value

A base data.frame with one row per row of newdata: row (its position in newdata), the id column for a two-level fit, and

- for "class": profile (0 is the noise component of a noise = TRUE fit) and its posterior, and for two-level fits group_class and group_posterior;
- for "posterior": those columns and one probability_profile_k column per profile (and probability_noise);
- for "density": log_density.

Groups

A two-level model classifies a row using its group: the group class is inferred from all of that group's rows, and it shifts the row's profile probabilities. The groups in newdata (the fit's id column) are treated as *new* groups, each classified from its own rows alone, which is the case of applying a model to a new cohort. A single-level fit, or a fit given id = NULL, treats every row as its own unit.

Conditions

latents_bad_data when newdata lacks an indicator or the id column, has a missing value the fit cannot handle (missing = "error"), a category the fit never saw, or a non-numeric continuous indicator.

Examples

```
activity <- c("browse", "lectures", "forum_read")
training <- subset(course_engagement, student <= 60)
fit <- mutilpa(training, activity, id = "student", n_profiles = 2,
               n_group_classes = 2, n_starts = 2, seed = 1)
new_students <- subset(course_engagement, student > 100)
head(predict(fit, new_students))
head(predict(fit, new_students, type = "density"))
```

```
print.mutilpa_enumeration
      Print latents results
```

Description

Print methods for the objects the package returns: fitted models, enumerations, bootstrap tests, covariate and transition models, pooled imputations, their summaries, result tables and plot lists. Each prints a compact overview; `get_results()` returns the tables themselves as data frames.

Usage

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

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

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

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

## S3 method for class 'mutilpa_covariates'
print(x, rows = 20L, ...)

## S3 method for class 'summary_mutilpa_covariates'
print(x, digits = 4L, rows = 10L, ...)

## S3 method for class 'summary_mutilpa_enumeration'
print(x, digits = 4L, rows = 10L, ...)

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

## S3 method for class 'summary_mutilpa_bootstrap_lrt'
print(x, digits = 4L, rows = 10L, ...)

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

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

## S3 method for class 'latents_table'
print(x, n = 20L, ...)
```

```
## S3 method for class 'multilpa'
print(x, rows = 20L, ...)

## S3 method for class 'summary_multilpa'
print(x, digits = 4L, rows = 10L, ...)

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

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

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

## S3 method for class 'latents_pooled'
print(x, digits = 4L, rows = 20L, ...)

## S3 method for class 'multilpa_transitions'
print(x, rows = 20L, ...)

## S3 method for class 'summary_multilpa_transitions'
print(x, digits = 4L, rows = 10L, ...)
```

Arguments

x	An object returned by a latents function, or the summary() of one.
...	Passed to the underlying data frame printing, or ignored.
digits	Number of significant digits printed.
rows	How many rows of each table to print. A longer table is shown to that depth, with its remaining row count and the get_results() call that returns it whole.
n	The most rows of a result table printed.

Value

x, invisibly. Called for the side effect of printing.

See Also

[get_results\(\)](#) for the tables, [latents-summary](#) and [latents-as-data-frame](#).

Examples

```
fit <- multilpa(course_engagement, c("browse", "lectures", "forum_read"),
               "student", n_profiles = 2, n_group_classes = 2,
               n_starts = 2, seed = 1)
print(fit, rows = 3)
```

prior_control

*Conjugate prior for Gaussian mixture estimation***Description**

Requests maximum a posteriori (MAP) estimation of the Gaussian profile means and covariances under the conjugate prior of Fraley and Raftery (2007), exactly as `mclust::priorControl()` does for `mclust::Mclust()`. Pass the result as `multilpa(prior = )`. The prior keeps covariances away from singularity, which is what lets a many-profile or many-indicator model be fitted where maximum likelihood degenerates.

Usage

```
prior_control(shrinkage = 0.01, mean = NULL, dof = NULL, scale = NULL)
```

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

Arguments

shrinkage	Non-negative prior precision factor for the means, kappa; mclust's default 0.01. Zero leaves the means unshrunk.
mean	Optional prior mean, one value per continuous indicator, in the indicators' own units. Naming one needs a positive shrinkage.
dof	Optional prior degrees of freedom, a single 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.
x	A latents_prior object.
...	Ignored.

Details

Every argument left NULL takes `mclust::defaultPrior()`'s value, computed from the indicators being fitted: the prior mean is their column 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.

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.

`print()` returns `x` invisibly.

Conditions

Raises `latents_bad_argument` for a negative or non-finite shrinkage, a mean with a zero shrinkage, or a non-positive dof or scale.

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()`, whose prior argument takes this.

Examples

```
prior_control()
prior_control(shrinkage = 0)
```

r3step	<i>Covariates predicting class membership, corrected for misclassification</i>
--------	--

Description

The R3STEP approach. A multinomial logit of class membership on covariates, fitted after the measurement model rather than alongside it, but treating the assigned class as an error-prone indicator of the true one with the error rates held fixed at what step one found. Regressing the modal class directly instead attenuates every coefficient, because some units are in the wrong class and the covariate cannot explain why.

Usage

```
r3step(
  x,
  data,
  covariates,
  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
)
```

Arguments

x	A covariate-free fitted <code>multilpa()</code> or <code>lta()</code> model. A fit that already uses membership covariates cannot be corrected by the fixed, unconditional classification-error matrix used here.
data	The data frame carrying the covariates, in the fit's row order. Shared fit columns are checked row by row; a frame with too little identifying information warns that alignment cannot be verified.

covariates	Character vector of numeric columns not used as measurement indicators in <code>x</code> . For <code>level = "groups"</code> each must be constant within a group.
level	"individuals" predicts profile membership, "groups" predicts group-class membership.
ci_level	Confidence level for the intervals.
vcov_type	"robust", the default, uses the sandwich clustered on the fit's groups, which is the honest choice when the covariates are measured on observations nested inside them, and matches the default of <code>three_step()</code> . "observed" uses the observed information; for the pooled regression it treats every observation as independent, and with <code>by_group_class = TRUE</code> it is the model-based variance of the two-level likelihood. "robust" needs more independent groups than the regression has coefficients, because the group score contributions sum to zero at the estimate and so span at most one dimension fewer than there are groups; with too few it is refused with <code>latents_too_few_groups</code> rather than reporting a variance that is singular, or, with a single group, numerically zero. "observed" remains available there.
adjust	Multiplicity correction applied across the covariate terms, passed to <code>stats::p.adjust()</code> . "BH" by default; "none" leaves the p-values uncorrected. The family is every covariate term of every non-reference class, which is the set of tests this call computes; the intercepts are not part of it.
by_group_class	FALSE, the default, fits one pooled multinomial logit with a single intercept per profile. TRUE fits the two-level form of the one-step model instead: every 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 alongside. Differences between groups that the covariates do not explain are then absorbed by the group classes rather than left in the residual, where the pooled regression attenuates the slopes and inflates their clustered standard errors. For <code>level = "individuals"</code> on a covariate-free <code>multilpa()</code> fit only; refused with <code>latents_bad_argument</code> otherwise.

Details

The error matrix is held fixed rather than estimated jointly, which is what makes this a three-step method and what keeps the covariates from reshaping the classes. On a `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.

Over 60 replications of a two-profile design with a true log-odds slope of 1.2 and intercept -0.3, this recovered the slope with a bias of -0.002 and the intercept with a bias of +0.006, where a logistic regression on the modal class was biased by -0.163 on the slope. The correction costs some precision: the standard deviation of the slope across replications was 0.127 against 0.100 for the naive fit. The nominal 95% intervals covered the true slope in 95% of those replications.

That coverage was measured where the profiles separate well. The error matrix is still treated as known rather than estimated, so intervals should be expected to run narrow where classification is poorer; check `get_results(x, "classification_errors")` before relying on them.

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 and `p_adjusted` carries the correction named by `adjust`, which is also recorded in the result's `adjust` attribute; it is NA on the intercept rows, which are not part of the tested family. A coefficient with zero standard error has an undefined Wald statistic and p-value, reported as `NA_real_` rather than infinity or zero. Coefficients are log odds against the final class, which is the reference, matching `multilpa(profile_covariates = )`; that class is named in the result's `reference_class` attribute, and the variance that was used in its `vcov_type` attribute. With `by_group_class = TRUE` the intercept rows are one per group class, named `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 `by_group_class` and the maximised `log_likelihoood` as attributes.

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.

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")
```

Description

Prints the estimates, then the classification diagnostics, then draws every plot the fit supports. A first look, not a substitute for the verbs: each section is what `summary()`, `diagnostics()` and `plot()` return, and the tables are reachable from those rather than from here.

Usage

```
report(
  x,
  data = NULL,
  plots = TRUE,
  by = c("profile", "overall"),
  rows = 10L,
  ...
)
```

Arguments

<code>x</code>	A fitted model of this package.
<code>data</code>	Optional. The data the model was fitted to; a fit carries the columns it was built from.
<code>plots</code>	TRUE, the default, draws the plots. FALSE prints only.
<code>by</code>	Passed to diagnostics() , and from there to the bivariate residuals: "profile", the default, assesses each profile separately, "overall" pools them.
<code>rows</code>	How many rows of each of the summary's tables to print, passed to <code>print(summary(x))</code> . A first look at a fit with thousands of observations would otherwise be mostly posteriors.
<code>...</code>	Nothing further is accepted. An argument this function cannot forward raises an error of class <code>latents_bad_argument</code> naming it, before anything has been printed, rather than being dropped.

Value

The fitted model, invisibly. Called for the printing and drawing.

What it prints

`summary()`, which is every table the fit can produce, then `descriptives()`, then the condensed reading of `diagnostics()`, then every plot view the fit supports. Each section is what that verb returns, and the tables are reached from [get_results\(\)](#) rather than from here.

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

Does the solution survive a different seed?

Description

A latent profile model is fitted by EM from random starts, so what comes back is a local maximum reached from one particular set of them. `n_starts` reports how many starts *within a single seed's stream* reached the best likelihood; it cannot say whether a different stream would have found a different mode. This verb refits the model under each of several seeds and reports what changed.

Usage

```
sensitivity(
  x,
  data = NULL,
  seeds = 1:10,
  n_starts = NULL,
  max_iter = NULL,
  tol = NULL,
  tolerance = NULL
)
```

Arguments

<code>x</code>	A fitted multilpa model to use as the reference. A <code>fit_staged()</code> result varies the second-stage starts while holding its first-stage measurement solution fixed. A directly fitted model with measurement blocks held is refused: its original free starting values are not retained, so replaying the same seed would not replay the same search.
<code>data</code>	Optional. The data the model was fitted to, in its original row order. A fit carries the columns it was built from, so this is only needed to override them. Shared identifier and indicator columns are checked against the fit before assignment agreement is calculated.
<code>seeds</code>	Seeds to refit under: at least two distinct whole numbers of either sign, as <code>set.seed()</code> accepts. Anything else raises <code>latents_bad_argument</code> . The reference fit's own seed may be among them, in which case that row reproduces it and is the arithmetic check that the refit is the same model.
<code>n_starts</code>	Starts per refit. Defaults to the number the reference fit used, so each seed gets the same search effort it did.
<code>max_iter, tol</code>	Passed to each refit; default to the reference fit's.
<code>tolerance</code>	Two log likelihoods within this of each other count as the same optimum. Defaults to the reference fit's own replication tolerance, which is what it used to decide whether its starts agreed.

Details

Three things are worth reading off the result. Whether every seed reached the same maximised log likelihood, which is what optimum counts. Whether the seeds that reached it converged. And how many observations were assigned to a different profile, which is what agreement measures, after the arbitrary profile labels of each refit have been matched to the reference fit's – two fits of the same mixture can be identical and still number their profiles differently, so comparing labels directly would report disagreement that is not there.

A seed whose refit fails contributes a row with NA estimates rather than being dropped silently, and a `latents_sensitivity_dropped` warning names how many failed, so the table is never quietly shorter than seeds.

Value

A base `data.frame`, one row per seed, with columns `seed`, `log_likelihood`, `converged`, `iterations`, `optimum` (1 for the best maximum found across the seeds, 2 for the next distinct one, and so on), `best` (whether this seed reached optimum 1) and `agreement` (the proportion of observations given the same profile as the reference fit, after label alignment). A failed refit has NA throughout except `seed`.

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 rather than the seed, 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)
```

simulate.latents_growth_mixture

Simulate outcomes from a growth mixture model

Description

Draws new outcomes on the fitted design: each person's class from their membership probabilities, their random effects from that class's covariance, and their outcomes around their own trajectory with the class's residual variance.

Usage

```
## S3 method for class 'latents_growth_mixture'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	A latents_growth_mixture fit.
nsim	Number of simulated outcome vectors.
seed	NULL or an integer seed; the caller's random state is restored afterwards.
...	Unused.

Value

A base data.frame with one row per fitted row and one column per simulation, sim_1, sim_2, ...

Examples

```
few_students <- subset(growth_scores, student <= 60)
small_fit <- mixture_regression(score ~ wave, few_students, n_classes = 2,
                              id = "student", class_level = "group",
                              random = "intercept", seed = 1)
head(simulate(small_fit, nsim = 2, seed = 1))

# All 300 students, with random intercepts and slopes
fit <- mixture_regression(score ~ wave, growth_scores, n_classes = 3,
                         id = "student", class_level = "group",
                         random = "wave", random_covariance = "equal",
                         n_starts = 2, seed = 1)
head(simulate(fit, nsim = 2, seed = 1))
```

```
simulate.latents_mixture_regression
```

Simulate outcomes from a mixture-of-regressions fit

Description

Draws class memberships from the fitted mixing model (group classes first in the two-level model, one class per group under class_level = "group"), then outcomes from each row's class regression, at the fitted rows' predictors.

Usage

```
## S3 method for class 'latents_mixture_regression'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	A fit from <code>mixture_regression()</code> .
nsim	Number of simulated outcome vectors.
seed	NULL or an integer; the caller's random state is restored.
...	Unused.

Value

A base `data.frame` with one row per fitted row and one column per simulation, `sim_1`, `sim_2`, ...; for a binomial fit with more than one trial the columns hold success counts, and for an ordinal fit the fitted categories in the outcome's own type (a factor with the fitted levels, or the category numbers).

Examples

```
fit <- mixture_regression(score ~ hours, data = study_hours, n_classes = 2,
                          n_starts = 3, seed = 1)
head(simulate(fit, nsim = 2, seed = 1))
```

srl	<i>Self-regulated learning scores simulated by a large language model</i>
-----	---

Description

Scores on five constructs of the Motivated Strategies for Learning Questionnaire (MSLQ) for 300 simulated respondents. Each score is the mean of the construct's Likert items, on a scale from 1 to 7. The responses were **generated by a large language model (GPT)**, not collected from students: they come from a study of how language models answer psychological surveys, and are used here because they are a well-structured, openly shareable cross-sectional example. They should not be read as evidence about students.

Usage

```
srl
```

Format

A data frame with 300 rows and 5 columns:

<code>cognitive_strategies</code>	cognitive strategy use
<code>intrinsic_value</code>	intrinsic value of the course
<code>self_efficacy</code>	self-efficacy
<code>self_regulation</code>	self-regulation
<code>test_anxiety</code>	test anxiety

Details

The five constructs are strongly correlated (about 0.8, except test anxiety), which makes the data a good illustration of how the choice of covariance structure shapes a latent profile analysis.

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	<i>Build starting values for a multilevel latent profile fit</i>
-----------------	--

Description

Returns the starting-value list `multilpa()` accepts, taken from a fitted model or from any object that already carries the parameter blocks, such as a retained reference solution. This exists so that callers never assemble the list by hand, and so that starting values are validated where they are built rather than deep inside the fitting loop.

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 whichever 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. Any other elements are ignored.
covariance	"auto" keeps covariances when the object carries them, "drop" always returns the diagonal parameterization, and "keep" requires covariances and fails when they are absent.
what	"all" returns the measurement and mixing blocks. "measurement" omits profile_probabilities and group_probabilities, which are sized for the model that produced them and must not be carried into a staged fit that changes the number of group classes. Pass the result as start alongside fixed in <code>multilpa()</code> .

Value

An object of class `multilpa_start`: the list `multilpa()` accepts as `start`, with elements `means`, `variances`, `profile_probabilities`, and `group_probabilities`, plus covariances when the full-covariance parameterization is returned and `response_probabilities` when the object carries a categorical measurement model. `what = "measurement"` drops `profile_probabilities` and `group_probabilities`. The Gaussian blocks are unnamed and indexed by position, matching what `multilpa()` expects of `start`; read them with `as.data.frame()`, which labels the positions, rather than out of the list. `response_probabilities` is the exception: the list keeps each indicator's name and each block keeps its category names, because those labels are what lets `multilpa()` attach a block to the item it was estimated for rather than to whichever item happens to sit in that position. A fit that names its categorical indicators in another order, or that has other categories, therefore refuses the `start` with `latents_bad_start` instead of reporting a likelihood for a measurement model it never held. The list payload is unchanged by the class, so a `multilpa_start` can be passed straight back as `start`.

See Also

`get_results()` for the values as tidy tables.

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

Experience-sampling data on what university students did between prompts when they were not studying. Prompts are nested within students, so the data suit a two-level latent class analysis: each prompt belongs to an activity profile, and students differ in how often their prompts fall in each profile.

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

The students answered six prompts a day for fourteen days. At a prompt where the student had not studied since the previous prompt, the questionnaire asked which of eight leisure activities they had done. The study asked these items only at non-study prompts, so the dataset contains those prompts alone. Each row is one answered non-study prompt with all eight activity items and all four affect ratings observed.

The dataset is a subset of the original study: 100 of its students, drawn with a fixed seed from the students with at least 15 such prompts, with every such prompt they answered. Each student contributes 15 to 51 prompts (median 24). The eight activity items are categorical indicators; the four affect ratings are continuous-scored and can be combined with them in a mixed measurement model. `worried` has a floor: 49.7% of its ratings are 1, so a profile of low-worry prompts can have zero variance on it and reach the `min_variance` bound.

Source

openESM dataset 0062, deposited by Neubauer and Schmiedek at Zenodo, [doi:10.5281/zenodo.17347974](https://doi.org/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](https://doi.org/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	<i>Study hours and quiz scores under two study strategies</i>
-------------	---

Description

A simulated dataset for `mixture_regression()`, built so that every nesting of the mixture regression has a model that generated some of its columns, and the truth is kept for checking recovery.

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. Not for fitting; kept to check recovery.
- student_type** Factor, "steady" or "erratic": the student's generating kind. Not for fitting; kept to check recovery.

Details

150 students each report six study weeks. In every week a student studies with one of two strategies, and the strategy decides how hours become a quiz score: a deep week gains about 4.5 points per hour from a base of 35, a surface week under one point per hour from a base of 55. Students are of two kinds that differ in their *mix* of strategies – steady students study deeply in about 85% of weeks, erratic ones in about 25% – and higher motivation makes a student more likely to be steady. More sleep makes a deep week more likely.

So score ~ hours is a two-class mixture at the level of the week (`class_level = "observation"`), whose class shares differ by student (the two-level model, `id = "student"`, `n_group_classes = 2`). passed depends on the student's kind, not the week's strategy, so it is a mixture at the level of the student (`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")
```

```
summary.multilpa_additive
```

Summarise latents results

Description

Summary methods for fitted models, enumerations, bootstrap tests, covariate and transition models, mixture regressions and pooled imputations. A summary collects the model's main tables and diagnostics; printing it shows each table to a set depth.

Usage

```
## S3 method for class 'multilpa_additive'
summary(object, level = 0.95, vcov_type = c("observed", "robust", "opg"), ...)

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

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

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

## S3 method for class 'latents_growth_mixture'
summary(object, level = 0.95, vcov_type = NULL, ...)

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

## S3 method for class 'latents_mixture_regression'
summary(object, level = 0.95, vcov_type = NULL, ...)

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

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

Arguments

object	An object returned by a latents function.
level	Confidence level for the reported intervals, strictly between zero and one.
vcov_type	"observed", "robust" or "opg", as for parameter_inference() . For mixture regressions and growth mixtures, NULL reuses the inference stored with the fit, or, when there is none, uses robust errors for a weighted fit and observed otherwise.
...	Ignored; present for compatibility with summary() .

Value

An object of class `summary_<class>` holding the summarised tables. Print it, or read any table with [get_results\(\)](#) or [as.data.frame\(\)](#).

See Also

[latents-print](#), [latents-as-data-frame](#), [get_results\(\)](#).

Examples

```
fit <- multilpa(course_engagement, c("browse", "lectures", "forum_read"),
               "student", n_profiles = 2, n_group_classes = 2,
               n_starts = 2, seed = 1)
print(summary(fit), rows = 3)
```

three_step

*Relate a latent class to an outcome it did not help define***Description**

The three-step approach: the measurement model is fitted first, units are assigned to classes, and only then is the outcome brought in, corrected for the fact that some assignments are wrong. Doing it in one step instead lets the outcome pull the classes toward itself; doing it naively by modal class instead attenuates every difference. This does neither.

Usage

```
three_step(
  x,
  data,
  outcome,
  level = c("individuals", "groups"),
  method = c("bch", "proportional", "modal"),
  ci_level = 0.95,
  contrast = c("none", "pairs"),
  adjust = c("BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "none"),
  vcov_type = c("cluster", "independent")
)
```

Arguments

x	A covariate-free fitted <code>multilpa()</code> or <code>lta()</code> model. A fit that already uses membership covariates cannot be corrected by the fixed, unconditional classification-error matrix used here.
data	The data frame carrying the outcome, in the fit's row order. Shared fit columns are checked row by row; a frame with too little identifying information warns that alignment cannot be verified.
outcome	Name of a numeric outcome column not used as a measurement indicator in x. For level = "groups" it must be constant within each group.
level	"individuals" relates the outcome to profiles, "groups" to group classes.
method	"bch" applies the Bolck-Croon-Hagenaars weights; "proportional" weights by the posteriors, which is simpler and attenuated; "modal" is the naive assignment, shown for comparison.
ci_level	Confidence level for the intervals.
contrast	"none" (the default) reports one outcome mean per class. "pairs" reports the difference between every pair of classes instead, which is the quantity a three-step analysis is usually run to test, with a statistic and a p-value against a difference of zero.
adjust	Multiplicity correction applied across the pairwise differences, passed to <code>stats::p.adjust()</code> . "BH" by default; "none" leaves the p-values uncorrected. Ignored for contrast = "none", which tests nothing.

`vcov_type` "cluster", the default, sums the influence contributions within the fit's groups and takes those groups as the independent units, which is the honest choice when the outcome is measured on observations nested inside them. "independent" treats every observation as its own independent unit instead. That ignores the nesting the model was fitted to and is only defensible when there is no nesting left to ignore, such as a fit with a single group; it is never substituted silently, and the choice is recorded in the result's `vcov_type` attribute.

Details

The correction assumes the outcome is independent of the assigned class given the true one, which is what makes a three-step method valid. On a `lta()` fit, `level = "individuals"` uses one row per occasion and relates the outcome to the profile at that occasion. The default group-clustered variance accounts for repeated occasions within a group. This is a marginal profile-outcome analysis; it does not estimate an outcome effect on transitions or a sequence-level outcome model.

A cluster-robust variance is the sum of one outer product per independent unit, and those contributions sum to zero at the estimate, so it has rank at most one less than the number of units. With a single group it is exactly zero and with as many groups as classes it is singular. Both are refused with `latents_too_few_groups` rather than reported as a very small standard error; `vcov_type = "independent"` is the labelled way to ask for the unclustered variance instead.

The error matrix is treated as known rather than estimated, so the intervals are optimistic and the amount is worth stating. Over 60 replications of a two-profile design with 8% misclassification and a true class difference of 10, "bch" recovered that difference with a bias of -0.13 against -1.28 for "modal" and -1.82 for "proportional", and its nominal 95% intervals covered the truth 85% of the time. Treat the point estimate as close to unbiased and the interval as somewhat too narrow; the other two methods are biased enough that their intervals covered the truth in none of the 60.

Value

A base data.frame. For `contrast = "none"` it has one row per class and the columns `level`, `method`, `class`, `estimate`, `standard_error`, `conf_low`, `conf_high` and `effective_n`. It carries no statistic or `p_value`, unlike `r3step()`: an outcome mean has no meaningful null value to be tested against, and a p-value for "this class's mean is zero" would answer a question nobody asked. The comparison that does have a null is the difference between two classes, and that is what `contrast = "pairs"` returns: one row per pair 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`, matching `r3step()`. A contrast with zero standard error has no defined test statistic or p-value; those columns are `NA_real_` rather than a spurious finite test.

Standard errors are cluster-robust in both shapes under the default `vcov_type = "cluster"`, taking the fit's groups as the independent units, so they remain honest when the outcome is measured on observations nested inside those groups; `vcov_type = "independent"` gives the unclustered variance instead. The variance that was used is recorded in the result's `vcov_type` attribute, and for `contrast = "pairs"` the correction applied to `p_adjusted` is recorded in its `adjust` attribute. A `contrast = "none"` table tests nothing, so it carries no `adjust` attribute.

References

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

Coefficients, covariance, intervals and likelihood of latents fits

Description

`coef()`, `vcov()`, `confint()`, `logLik()` and `nobs()` for multilevel profile and class models (`multilpa()`), covariate models, latent transition models and, for `vcov()`, pooled imputations. Mixture regressions, growth mixtures and the additive, cross-level and general transition families document the same methods with their `get_results()` page.

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_covariates'
coef(object, scale = c("natural", "unconstrained"), ...)
```

```

## S3 method for class 'multilpa_covariates'
confint(object, parm, level = 0.95, data = NULL, ...)

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

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

## 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, ...)

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

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

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

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

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

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

```

Arguments

object	A fitted model: from multilpa() (class <code>multilpa</code>), a covariate model (<code>multilpa_covariates</code>), a latent transition model (<code>multilpa_transitions</code>), or pooled imputations (<code>latents_pooled</code> , <code>vcov()</code> only).
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), which round-trips exactly; supplying it is the stronger check that the caller still holds that frame.
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 in the units <code>coef()</code> reports: means and variances in the indicators' units, probabilities as probabilities, with the covariance carried from the estimation scale by the delta method. Unconstrained is the scale the model is estimated on: 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.
boundary	"error" or "fix", as for <code>parameter_inference()</code> : "fix" holds response probabilities on their bound, whose rows and columns are then zero (covariate models).
...	For <code>vcov()</code> and <code>confint()</code> of a multilpa fit, and <code>confint()</code> of a covariate model, passed to <code>parameter_inference()</code> (for example <code>method = "bootstrap"</code> , <code>vcov_type = "robust"</code> or <code>step</code>). Ignored otherwise.
parm	Optional coefficient names or indices; defaults to every estimated coefficient. Naming a coefficient that fixed held raises <code>latents_held_parameter</code> , because a held value has no interval.
level	Confidence level strictly between zero and one.

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 groups, the independent units of the likelihood.

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, 9
 - growth_schools, 48
 - growth_scores, 50
 - srl, 120
 - student_esm, 122
 - study_hours, 124
- as.data.frame(), 4, 6, 17–19, 21, 32, 66, 105, 122, 126
- as.data.frame.latents_family_enumeration
 - (get_results.latents_family_enumeration), 33
- as.data.frame.latents_growth_mixture
 - (get_results.latents_growth_mixture), 34
- as.data.frame.latents_mixture_regression
 - (get_results.latents_mixture_regression), 36
- as.data.frame.latents_pooled
 - (get_results.latents_pooled), 39
- as.data.frame.latents_regression_enumeration
 - (enumerate_regressions), 20
- as.data.frame.latents_table
 - (as.data.frame.mutilpa), 3
- as.data.frame.latents_transition_enumeration
 - (get_results.latents_transition_enumeration), 40
- as.data.frame.mutilpa, 3
- as.data.frame.mutilpa_additive
 - (get_results.mutilpa_additive), 41
- as.data.frame.mutilpa_bootstrap_lrt
 - (as.data.frame.mutilpa), 3
- as.data.frame.mutilpa_covariates
 - (as.data.frame.mutilpa), 3
- as.data.frame.mutilpa_cross_level
 - (get_results.mutilpa_cross_level), 43
- as.data.frame.mutilpa_diagnostics
 - (diagnostics), 14
- as.data.frame.mutilpa_enumeration
 - (as.data.frame.mutilpa), 3
- as.data.frame.mutilpa_lta
 - (get_results.mutilpa_lta), 45
- as.data.frame.mutilpa_start
 - (as.data.frame.mutilpa), 3
- as.data.frame.mutilpa_transitions
 - (as.data.frame.mutilpa), 3
- as.data.frame.summary_mutilpa
 - (as.data.frame.mutilpa), 3
- as.data.frame.summary_mutilpa_additive
 - (as.data.frame.mutilpa), 3
- as.data.frame.summary_mutilpa_bootstrap_lrt
 - (as.data.frame.mutilpa), 3
- as.data.frame.summary_mutilpa_covariates
 - (as.data.frame.mutilpa), 3
- as.data.frame.summary_mutilpa_enumeration
 - (as.data.frame.mutilpa), 3
- as.data.frame.summary_mutilpa_transitions
 - (as.data.frame.mutilpa), 3
- bootstrap_lrt, 5
- bootstrap_lrt(), 20, 52–54, 75, 77, 78
- candidate_fit, 7
- candidate_fit(), 17–19, 33
- coef(), 66, 86
- coef.latents_growth_mixture
 - (get_results.latents_growth_mixture), 34
- coef.latents_mixture_regression
 - (get_results.latents_mixture_regression), 36
- coef.mutilpa
 - (vcov.mutilpa_covariates), 129
- coef.mutilpa_additive
 - (get_results.mutilpa_additive), 41

coef.mutilpa_covariates
 (vcov.mutilpa_covariates), 129
 coef.mutilpa_lta
 (get_results.mutilpa_lta), 45
 coef.mutilpa_transitions
 (vcov.mutilpa_covariates), 129
 compare_models, 8
 compare_models(), 50, 87
 confint(), 54, 66
 confint.latents_growth_mixture
 (get_results.latents_growth_mixture), 34
 confint.latents_mixture_regression
 (get_results.latents_mixture_regression), 36
 confint.mutilpa
 (vcov.mutilpa_covariates), 129
 confint.mutilpa_additive
 (get_results.mutilpa_additive), 41
 confint.mutilpa_covariates
 (vcov.mutilpa_covariates), 129
 confint.mutilpa_transitions
 (parameter_inference.mutilpa_transitions), 85
 course_engagement, 9

 data.frame(), 4
 descriptives, 11
 descriptives(), 15, 54, 116
 diagnostics, 14
 diagnostics(), 13, 30, 32, 115, 116

 enumerate_classes, 15
 enumerate_classes(), 7, 8, 10, 18, 19, 55, 74, 101, 118
 enumerate_lca, 18
 enumerate_lca(), 16, 20, 57, 101
 enumerate_lpa, 19
 enumerate_lpa(), 16, 18, 58, 101
 enumerate_regressions, 20
 enumerate_regressions(), 68, 125

 fit_staged, 22
 fit_staged(), 6, 29, 51, 53, 73, 75, 83, 84, 117

 get_group_tna, 25
 get_group_tna(), 48, 54, 103

 get_results, 26
 get_results(), 3, 5, 15, 24, 62, 63, 66, 110, 111, 116, 122, 126, 129
 get_results.latents_family_enumeration, 33
 get_results.latents_family_enumeration(), 17
 get_results.latents_growth_mixture, 34
 get_results.latents_growth_mixture(), 49, 66, 107
 get_results.latents_mixture_regression, 36
 get_results.latents_mixture_regression(), 66
 get_results.latents_pooled, 39
 get_results.latents_regression_enumeration
 (enumerate_regressions), 20
 get_results.latents_transition_enumeration, 40
 get_results.latents_transition_enumeration(), 17
 get_results.mutilpa_additive, 41
 get_results.mutilpa_additive(), 55, 76
 get_results.mutilpa_cross_level, 43
 get_results.mutilpa_cross_level(), 77
 get_results.mutilpa_lta, 45
 get_results.mutilpa_lta(), 62
 get_tna, 47
 get_tna(), 25, 54, 103
 growth_schools, 48
 growth_scores, 50

 latents-as-data-frame, 111, 126
 latents-as-data-frame
 (as.data.frame.mutilpa), 3
 latents-conditions, 51
 latents-model-methods
 (vcov.mutilpa_covariates), 129
 latents-print, 5, 126
 latents-print
 (print.mutilpa_enumeration), 110
 latents-summary, 5, 111
 latents-summary
 (summary.mutilpa_additive), 125
 lca, 56
 lca(), 18, 58
 logLik(), 66

- logLik.latents_growth_mixture
(get_results.latents_growth_mixture),
34
- logLik.latents_mixture_regression
(get_results.latents_mixture_regression),
36
- logLik.mutilpa
(vcov.mutilpa_covariates), 129
- logLik.mutilpa_additive
(get_results.mutilpa_additive),
41
- logLik.mutilpa_covariates
(vcov.mutilpa_covariates), 129
- logLik.mutilpa_cross_level
(get_results.mutilpa_cross_level),
43
- logLik.mutilpa_lta
(get_results.mutilpa_lta), 45
- logLik.mutilpa_transitions
(vcov.mutilpa_covariates), 129
- lpa, 57
- lpa(), 20, 57
- lta, 59
- lta(), 11, 17, 25, 45, 47, 48, 54, 77, 82, 85,
86, 113, 114, 127, 128
- mixture_regression, 63
- mixture_regression(), 8, 20, 21, 37, 49, 50,
55, 91, 108, 120, 124, 125
- mutilca, 68
- mutilca(), 18, 25, 48, 56, 57, 78, 109
- mutilpa, 70
- mutilpa(), 5, 8, 11, 13, 16, 17, 19, 24, 25,
48, 55–59, 62, 63, 69, 82, 102, 104,
105, 109, 113, 114, 121, 122, 124,
127, 129, 130
- nobs(), 66
- nobs.latents_growth_mixture
(get_results.latents_growth_mixture),
34
- nobs.latents_mixture_regression
(get_results.latents_mixture_regression),
36
- nobs.mutilpa
(vcov.mutilpa_covariates), 129
- nobs.mutilpa_additive
(get_results.mutilpa_additive),
41
- nobs.mutilpa_covariates
(vcov.mutilpa_covariates), 129
- nobs.mutilpa_cross_level
(get_results.mutilpa_cross_level),
43
- nobs.mutilpa_lta
(get_results.mutilpa_lta), 45
- nobs.mutilpa_transitions
(vcov.mutilpa_covariates), 129
- parameter_inference
(parameter_inference.mutilpa_additive),
79
- parameter_inference(), 24, 29, 30, 52–54,
74–76, 78, 83, 86, 103, 105, 118,
126, 130, 131
- parameter_inference.mutilpa_additive,
79
- parameter_inference.mutilpa_transitions,
85
- plot(), 6, 17–19, 66, 115
- plot.latents_comparison, 87
- plot.latents_family_enumeration, 88
- plot.latents_growth_mixture, 89
- plot.latents_growth_mixture(), 66, 90,
91
- plot.latents_mixture_regression, 90
- plot.latents_pooled, 91
- plot.latents_transition_enumeration,
92
- plot.mutilpa, 93
- plot.mutilpa(), 98, 99, 103
- plot.mutilpa_additive, 96
- plot.mutilpa_bootstrap_lrt, 97
- plot.mutilpa_covariates, 98
- plot.mutilpa_cross_level, 99
- plot.mutilpa_diagnostics
(diagnostics), 14
- plot.mutilpa_enumeration, 100
- plot.mutilpa_lta, 102
- plot.mutilpa_transitions, 102
- plot_views, 104
- plot_views(), 94, 95, 116
- pool_imputations, 104
- pool_imputations(), 39, 54
- predict(), 66, 77
- predict.latents_growth_mixture, 107
- predict.latents_mixture_regression,
107

```

predict.mutilpa, 108
print.latents_family_enumeration
  (get_results.latents_family_enumeration), 110
  33
print.latents_growth_mixture
  (print.mutilpa_enumeration), 110
print.latents_mixture_regression
  (print.mutilpa_enumeration), 110
print.latents_plots
  (print.mutilpa_enumeration), 110
print.latents_pooled
  (print.mutilpa_enumeration), 110
print.latents_prior(prior_control), 112
print.latents_regression_enumeration
  (enumerate_regressions), 20
print.latents_table
  (print.mutilpa_enumeration), 110
print.latents_transition_enumeration
  (get_results.latents_transition_enumeration), 40
print.mutilpa
  (print.mutilpa_enumeration), 110
print.mutilpa_additive
  (print.mutilpa_enumeration), 110
print.mutilpa_bootstrap_lrt
  (print.mutilpa_enumeration), 110
print.mutilpa_covariates
  (print.mutilpa_enumeration), 110
print.mutilpa_cross_level
  (get_results.mutilpa_cross_level), 43
print.mutilpa_diagnostics
  (diagnostics), 14
print.mutilpa_enumeration, 110
print.mutilpa_lta
  (get_results.mutilpa_lta), 45
print.mutilpa_start
  (print.mutilpa_enumeration), 110
print.mutilpa_transitions
  (print.mutilpa_enumeration), 110
print.summary_latents_growth_mixture
  (print.mutilpa_enumeration), 110
print.summary_latents_mixture_regression
  (print.mutilpa_enumeration), 110
print.summary_mutilpa
  (print.mutilpa_enumeration), 110
print.summary_mutilpa_additive
  (print.mutilpa_enumeration), 110
print.summary_mutilpa_bootstrap_lrt
  (print.mutilpa_enumeration), 110
print.summary_mutilpa_covariates
  (print.mutilpa_enumeration), 110
print.summary_mutilpa_enumeration
  (print.mutilpa_enumeration), 110
print.summary_mutilpa_transitions
  (print.mutilpa_enumeration), 110
prior_control, 112
prior_control(), 75
r3step, 113
r3step(), 10, 29, 52–54, 71, 75, 77, 128
report, 115
sensitivity, 117
sensitivity(), 54, 55
simulate.latents_growth_mixture, 118
simulate.latents_mixture_regression, 119
srl, 120
starting_values, 121
starting_values(), 24, 53, 75
stats::p.adjust(), 81, 86, 114, 127
stats::p.adjust.methods, 42
student_esm, 122
study_hours, 124
summary(), 6, 15, 17–19, 32, 66, 115, 116, 126
summary.latents_family_enumeration
  (get_results.latents_family_enumeration),

```

[33](#)
 summary.latents_growth_mixture
 (summary.mutilpa_additive),
 [125](#)
 summary.latents_mixture_regression
 (summary.mutilpa_additive),
 [125](#)
 summary.latents_pooled
 (summary.mutilpa_additive),
 [125](#)
 summary.latents_transition_enumeration
 (get_results.latents_transition_enumeration), (parameter_inference.mutilpa_transitions),
 [40](#)
 summary.mutilpa
 (summary.mutilpa_additive),
 [125](#)
 summary.mutilpa_additive, [125](#)
 summary.mutilpa_bootstrap_lrt
 (summary.mutilpa_additive),
 [125](#)
 summary.mutilpa_covariates
 (summary.mutilpa_additive),
 [125](#)
 summary.mutilpa_cross_level
 (get_results.mutilpa_cross_level),
 [43](#)
 summary.mutilpa_enumeration
 (summary.mutilpa_additive),
 [125](#)
 summary.mutilpa_enumeration(), [17](#)
 summary.mutilpa_lta
 (get_results.mutilpa_lta), [45](#)
 summary.mutilpa_transitions
 (summary.mutilpa_additive),
 [125](#)

 three_step, [127](#)
 three_step(), [28](#), [29](#), [52–54](#), [71](#), [75](#), [77](#), [114](#),
 [115](#)
 tna::tna(), [25](#), [47](#), [48](#)

 vcov(), [54](#), [66](#)
 vcov.latents_growth_mixture
 (get_results.latents_growth_mixture),
 [34](#)
 vcov.latents_mixture_regression
 (get_results.latents_mixture_regression),
 [36](#)

 vcov.latents_pooled
 (vcov.mutilpa_covariates), [129](#)
 vcov.mutilpa
 (vcov.mutilpa_covariates), [129](#)
 vcov.mutilpa_additive
 (get_results.mutilpa_additive),
 [41](#)
 vcov.mutilpa_covariates, [129](#)
 vcov.mutilpa_lta
 (get_results.mutilpa_lta), [45](#)
 vcov.mutilpa_transitions
 (parameter_inference.mutilpa_transitions),
 [85](#)