Package: MAIHDA (via r-universe)

October 28, 2026

Type Package

Title Multilevel Analysis of Individual Heterogeneity and Discriminatory Accuracy

Version 0.3.0

Description Tools for Multilevel Analysis of Individual Heterogeneity and Discriminatory Accuracy (MAIHDA) for intersectional inequality research. Methods are described in Merlo (2018) <doi:10.1016/j.socscimed.2017.12.026> and Evans et al. (2018) <doi:10.1016/j.socscimed.2017.11.011>. The package creates intersectional strata, fits multilevel MAIHDA models, estimates variance partition coefficients, proportional change in variance, stratum effects, and discriminatory-accuracy summaries, and provides diagnostic and presentation plots.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports lme4 (>= 1.1-27), Matrix, reformulas, cli, ggplot2 (>= 3.4.0), dplyr (>= 1.0.0), tidyr (>= 1.1.0), generics, stats, rlang (>= 0.4.0), patchwork, ggrepel, tidyselect, tibble

Suggests shiny, bslib (>= 0.7.0), DT, future, promises, plotly, haven, shinyjs, shinycssloaders, brms (>= 2.15.0), WeMix (>= 4.0.0), ordinal, pbkrtest, testthat (>= 3.0.0), knitr, MASS, rmarkdown

VignetteBuilder knitr

URL https://github.com/hdbt/MAIHDA, https://hdbt.github.io/MAIHDA/

BugReports https://github.com/hdbt/MAIHDA/issues

RoxygenNote 7.3.3

NeedsCompilation no

Author Hamid Bulut [aut, cre]

Maintainer Hamid Bulut <me@hamidbulut.com>

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

Date/Publication 2026-10-28 17:10:02 UTC

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

RemoteRef HEAD

RemoteSha 2ce381b9426fe0ec83070e2074a7e24dccc263a4

Contents

calculate_pcv . . . 3
compare_maihda . . . 7
compare_maihda_groups . . . 9
fit_maihda . . . 13
maihda . . . 23
maihda_auc . . . 31
maihda_country_data . . . 32
maihda_cumulative . . . 34
maihda_describe . . . 35
maihda_discriminatory_accuracy . . . 38
maihda_glance . . . 41
maihda_health_data . . . 42
maihda_ic . . . 43
maihda_interactions . . . 45
maihda_long_data . . . 50
maihda_mor . . . 51
maihda_proportional_odds_test . . . 52
maihda_sim_data . . . 54
maihda_sparse_data . . . 55
maihda_table . . . 56
maihda_tidiers . . . 58
maihda_upset_size . . . 60
maihda_vpc_response . . . 61
make_strata . . . 63
pcv_importance . . . 65
plot.maihda_analysis . . . 69
plot.maihda_comparison . . . 71
plot.maihda_describe . . . 72
plot.maihda_group_comparison . . . 73
plot.maihda_model . . . 74
plot.maihda_pcv_importance . . . 77
plot_prediction_deviation_panels . . . 78
predict.maihda_model . . . 80
predict_maihda . . . 81
print.maihda_analysis . . . 83
print.maihda_describe . . . 84
print.maihda_group_comparison . . . 84
print.maihda_ic . . . 85

print.maihda_interactions 85
print.maihda_long_pcv 86
print.maihda_model 86
print.maihda_pcv_importance 87
print.maihda_stepwise 87
print.maihda_strata 88
print.maihda_summary 88
print.maihda_table 89
print.pcv_result 89
run_maihda_app 90
stepwise_pcv 90
summary.maihda_analysis 93
summary.maihda_model 94
theme_maihda 100

Index **101**

`calculate_pcv` *Calculate Proportional Change in Between-Stratum Variance (PCV)*

Description

Calculates the proportional change in between-stratum variance (PCV) between two MAIHDA models. The PCV measures how much the between-stratum variance changes when moving from one model to another, and is calculated as: PCV = (Var_model1 - Var_model2) / Var_model1.

Usage

```
calculate_pcv(
  model1,
  model2,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  estimation = c("fitted", "ML", "REML")
)
```

Arguments

`model1`	A maihda_model object from `fit_maihda()`. This is the reference model (typically a simpler or baseline model).
`model2`	A maihda_model object from `fit_maihda()`. This is the comparison model (typically a more complex model with additional predictors).
`bootstrap`	Logical indicating whether to compute bootstrap confidence intervals for the PCV. Default is FALSE. **lme4 engine only**: the parametric bootstrap relies on lme4's `simulate()`/`refit()`, so for the brms, wemix, and ordinal engines the PCV is reported as a point estimate and `bootstrap = TRUE` is an error (see Details).

`n_boot`	Number of bootstrap samples if bootstrap = TRUE. Default is 1000. A value below about 200 warns that the interval's tail endpoints are unstable (the hard minimum is 10).
`conf_level`	Confidence level for bootstrap intervals. Default is 0.95.
`estimation`	Which between-stratum variances are compared: `"fitted"` (default), `"ML"` or `"REML"`. `"fitted"` uses each model's own estimate (REML for a Gaussian `lmer` fit, maximum likelihood for a `glmer` fit). `"ML"` refits REML `lmer` fits by maximum likelihood first, which overstates the PCV when strata are few. `"REML"` compares REML-type variances for binomial and Poisson `glmer` fits, whose maximum-likelihood PCV is overstated in the same way. See Details.

Details

The PCV is the proportional change in between-stratum variance when moving from model1 to model2: a positive value means model2 has lower between-stratum variance, a negative value means higher. It is the share of model1's between-stratum variance *explained* by model2 only in the canonical nested case, where model2 adds fixed-effect predictors to model1 on the same outcome, analytic sample and strata. The function does not require nesting, so for non-nested models the PCV is simply a model-dependent difference in variance, not an explained proportion.

REML vs ML (the `estimation` argument). `estimation` sets which between-stratum variances are compared:

`"fitted"` **(default)** each model's own estimate – the REML estimate for a Gaussian `lmer` fit, the variance `summary.maihda_model` reports.

`"ML"` REML `lmer` fits are refitted by maximum likelihood (`refitML`) first, and the bootstrap interval is computed on the same basis. Use it to match an analysis fitted by maximum likelihood.

`"REML"` REML-type variances. REML `lmer` fits are used as fitted. For a binomial or Poisson `glmer` fit whose only random effect is the stratum intercept, the between-stratum variance is re-estimated with the fixed effects integrated out, as REML does for a Gaussian model (see below).

The PCV is a ratio of two variance estimates and uses no likelihood value, so it does not need maximum-likelihood fits (likelihood-ratio tests and information criteria do; see `maihda_ic`). REML estimates each model's between-stratum variance with little bias. Maximum likelihood makes no allowance for the degrees of freedom spent on the fixed effects, so it underestimates that variance, and by more in the adjusted model than in the null model: the PCV is *overstated*, most of all with few strata. In a balanced design whose fixed effects are constant within strata, ML multiplies REML's estimate of the variance of the stratum means, $\sigma_u^2 + \sigma_e^2/n$, by $(J - p)/J$ (J strata, p fixed-effect coefficients, neither estimate at zero): 23/24 for the null model of a 2 x 3 x 4 design but 17/24 for the adjusted one. The bootstrap interval inherits the bias, because it simulates from and refits on the same basis.

`"fitted"` and `"ML"` differ for Gaussian `lmer` fits only. `glmer` and the wemix and ordinal engines estimate variances by maximum likelihood, so under either setting their PCV is overstated in the same way: most with few strata, and more when strata are small or uneven in size, because the shortfall applies to the between-stratum variance plus the sampling variance of a stratum's own estimate. `print()` notes when the compared variances are maximum-likelihood estimates.

`estimation = "REML"` corrects that shortfall for binomial and Poisson `glmer` fits. Each variance minimises the Laplace-approximated deviance evaluated at the joint mode of the fixed and random

effects (lme4's nAGQ = 0 deviance) plus the log-determinant of the fixed-effect information, which integrates the fixed effects out; for the logit and log links this is the criterion glmmTMB(REML = TRUE) minimises. The models are not refitted, and the as-fitted values are returned alongside (pcv_fitted, var_model1_fitted, var_model2_fitted).

Use it when strata are few but each holds enough observations for its own estimate to be precise: the PCV then has little bias, though with very few strata it stays imprecise. When the strata are small as well, neither basis is dependable. A variance estimate cannot fall below zero, so with that much noise the REML-type estimate averages too high, most in the adjusted model: the PCV then varies widely from sample to sample and can come out far too low, even negative, where the maximum-likelihood PCV stays too high.

estimation = "REML" is an error for the wemix and ordinal engines, for other glmer families, for a fit with a further random effect such as a context, for an lmer fit made by maximum likelihood, and with bootstrap = TRUE on a glmer fit.

A brms fit is a Bayesian posterior: "ML" and "REML" perform no refit and estimation_used is "posterior". Single-model VPC/ICC summaries always use the model as fitted.

Latent-scale families and rescaling. For families whose level-1 variance is a fixed latent-scale constant – binomial/Bernoulli ($\pi^2/3$ logit, 1 probit) and the cumulative (ordinal) model – the linear predictor is identified only up to scale. Adding predictors that explain *within-stratum* (individual-level) variation cannot shrink that fixed level-1 variance; the latent scale stretches instead, inflating the coefficients and the between-stratum variance alike (Bauer 2009; Mood 2010). Part of a null-vs-adjusted change in the between-stratum variance is then rescaling rather than genuinely explained variance, so latent-scale PCVs tend to be understated and can turn negative on this account alone. The canonical MAIHDA adjusted model – which adds the stratum dimensions' main effects, constant *within* each stratum – is largely unaffected, but the caveat is first-order whenever an added predictor varies within strata (an individual-level covariate, as in the stepwise_pcv steps that add one). The count families' level-1 variance is not a fixed constant, but as with any non-identity link the same non-collapsibility logic applies in attenuated form. Gaussian identity-link PCVs are not subject to this.

When bootstrap = TRUE, the function uses a parametric bootstrap: it simulates new responses from model2 and refits both models with lme4::refit() for each simulated response to obtain confidence intervals for the PCV estimate. For negative-binomial models (glmer.nb) refit() holds the dispersion parameter theta fixed at its original estimate, so the interval is conditional on the estimated theta.

A bootstrap draw whose *null-model* between-stratum variance lands on the zero boundary has no defined PCV (the denominator is zero); such draws are excluded, so the percentile interval is *conditional on estimating a positive null variance*. Whenever any draws hit the boundary the function warns, reports the count as n_boot_boundary on the result, and print() repeats the caveat – a sizeable boundary share signals weak between-stratum variation, and the PCV itself is then fragile.

Bootstrap draws whose refit optimiser reports non-convergence are kept in the interval (they still have a defined PCV) and counted in n_boot_nonconverged, which print() shows. When more than half the contributing draws did not converge the interval is still returned, but with a warning and interval_reliable = FALSE; treat it as indicative only and check for singular or failing fits.

The bootstrap is available for the lme4 engine only. For the other engines the PCV is a *point estimate*: a brms fit's posterior credible interval (reported by summary.maihda_model) covers a single fit's VPC/ICC, not the PCV, which compares two separately fitted models – no posterior interval for the PCV itself is computed – and a design-based (wemix) interval would require replicate weights.

brms fits. For two brms fits the variances compared are the posterior medians of each model's between-stratum variance, the values the variance-components table of summary.maihda_model shows. The posterior of a between-stratum variance is right-skewed when strata are few, and more so in the adjusted model, which spends degrees of freedom on the main effects. The PCV is therefore *understated* with few strata – the opposite direction to maximum likelihood – and with very few strata it can be negative where the true value is positive. The understatement shrinks as strata are added.

Value

A list containing:

pcv	The estimated proportional change in variance
pvc	Deprecated duplicate of pcv; reading it warns, and it will be removed in a future release
var_model1	Between-stratum variance from model1
var_model2	Between-stratum variance from model2
estimation	The variance-estimation basis requested ("fitted", "ML" or "REML")
estimation_used	The basis actually used: "fitted", "ML", "mixed", "REML" or "posterior". "mixed" arises under estimation = "ML" when a model keeps its REML fit – its between-stratum variance is on the boundary so the ML refit is skipped, or refitML failed – so the comparison is partly REML rather than the pure ML one requested. "REML" is reported when the REML-type variances of a glmer fit were compared; estimation = "REML" on REML lmer fits uses them as fitted and reports "fitted". "posterior" is reported for a brms comparison (a Bayesian posterior, on which "ML" and "REML" are no-ops). print() states the basis
ml_variances	Logical; TRUE when the compared variances are maximum-likelihood estimates (a glmer, wemix or ordinal fit, or an lmer fit made or refitted by maximum likelihood). print() notes it under the default estimation
reml_available	Logical; TRUE when estimation = "REML" would compare REML-type variances for these two models (binomial or Poisson glmer fits it accepts)
pcv_fitted, var_model1_fitted, var_model2_fitted	Present when estimation_used is "REML": the PCV and the two variances as fitted by maximum likelihood, the values estimation = "fitted" returns
adjusted_at_boundary	Logical; TRUE when model2's between-stratum variance is on the singularity boundary, so the PCV is pinned near 1 (100%) and its interval/SE are unreliable – consistent with genuinely additive strata as well as a degenerate fit (print() states this)
ml_refit_failed	Logical; TRUE when estimation = "ML" was requested but refitML failed for a model, so its REML fit was used instead (the comparison is then not on a pure ML basis)
ci_lower	Lower bound of confidence interval (if bootstrap = TRUE)

`ci_upper` Upper bound of confidence interval (if bootstrap = TRUE)

`bootstrap` Logical indicating if bootstrap was used

`n_boot_nonconverged`
Number of contributing bootstrap draws whose refit optimiser did not converge (`optinfo$conv$opt != 0`); these are retained in the interval and counted, so `n_boot_ok` does not imply convergence

`interval_reliable`
Logical (bootstrap only); `FALSE` when more than half the contributing draws did not converge, in which case the interval is still returned but flagged unreliable (see Details)

References

Bauer, D. J. (2009). A note on comparing the estimates of models for cluster-correlated or longitudinal data with binary or ordinal outcomes. *Psychometrika*, 74(1), 97-105.

Mood, C. (2010). Logistic regression: why we cannot do what we think we can do, and what we can do about it. *European Sociological Review*, 26(1), 67-82.

See Also

`stepwise_pcv` for the sequential (one-variable-at-a-time) PCV, and `maihda` which computes the canonical null-vs-adjusted PCV automatically.

Examples

```
# Create strata and fit two models
strata_result <- make_strata(maihda_sim_data, c("gender", "race"))
model1 <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata_result$data)
model2 <- fit_maihda(health_outcome ~ age + gender + (1 | stratum), data = strata_result$data)

# Calculate PCV without bootstrap
pcv_result <- calculate_pcv(model1, model2)
print(pcv_result$pcv)

# Calculate PCV with bootstrap CI
# pcv_boot <- calculate_pcv(model1, model2, bootstrap = TRUE, n_boot = 500)
# print(pcv_boot)
```

`compare_maihda` *Compare MAIHDA Models*

Description

Compares variance partition coefficients (VPC/ICC) across multiple MAIHDA models, with optional bootstrap confidence intervals.

Usage

```
compare_maihda(
  ...,
  model_names = NULL,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  ic = TRUE
)
```

Arguments

`...`	Multiple maihda_model objects to compare.
`model_names`	Optional character vector of names for the models.
`bootstrap`	Logical; for **lme4** models, compute parametric-bootstrap VPC confidence intervals. Default FALSE. It does not apply to **brms** models, which always return a posterior credible interval (so passing `bootstrap = TRUE` with brms models errors) – their interval is included regardless.
`n_boot`	Number of bootstrap samples if bootstrap = TRUE. Default is 1000.
`conf_level`	Confidence level for the VPC interval (lme4 bootstrap CI or brms credible interval). Default is 0.95.
`ic`	Logical; append relative-fit information criteria to the table for comparing model *structures*: `AIC/BIC` for the likelihood engines (lme4, ordinal) and `WAIC/LOOIC` for brms (see `maihda_ic`). Default TRUE. REML `lmer` fits are refitted with ML so AIC/BIC are comparable across different fixed effects. Set FALSE for the lean VPC-only table.

Details

VPCs are only directly comparable when the models share an outcome, family/link, analytic sample, and strata – the canonical use is nested models (e.g. null vs covariate-adjusted) on the *same* data and strata, to show how the VPC attenuates. If the supplied models differ in any of these, `compare_maihda()` still returns the table but issues a single warning, because the VPCs are then not directly comparable. The appended information criteria follow the rules of `maihda_ic` instead: they need the same outcome, analytic sample and weights but not the same strata, and Poisson and negative-binomial fits of the same counts, or binomial or cumulative fits differing in their link, have comparable criteria although their VPCs are not comparable. In addition, when the appended criteria mix scales – likelihood `AIC/BIC` (lme4/ordinal) shown alongside Bayesian `WAIC/LOOIC` (brms), which can happen for a same-family lme4-vs-brms comparison that the family/link check does not flag – `compare_maihda()` warns, because those criteria are on different scales and are not comparable to each other.

Value

A `maihda_comparison` data frame of VPC/ICC by model. Interval columns (`ci_lower`/`ci_upper`) are included when any model supplies an interval – an lme4 bootstrap CI or a brms posterior credible interval. When `ic = TRUE`, information-criteria columns (`AIC/BIC` or `WAIC/LOOIC`, whichever apply) are appended.

Examples

```
# Canonical use: nested models on the SAME data and strata (null vs adjusted)
strata <- make_strata(maihda_sim_data, vars = c("gender", "race"))

null_model <- fit_maihda(health_outcome ~ 1 + (1 | stratum), data = strata$data)
adj_model  <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata$data)

# Compare without bootstrap
comparison <- compare_maihda(null_model, adj_model,
                             model_names = c("Null", "Adjusted"))

# Compare with bootstrap CI
comparison_boot <- compare_maihda(null_model, adj_model,
                                  model_names = c("Null", "Adjusted"),
                                  bootstrap = TRUE, n_boot = 500)
```

---

compare_maihda_groups *Compare MAIHDA Metrics Across Levels of a Grouping Variable*

Description

Fits a separate random-intercept MAIHDA model (intercept-only *random* effects; any fixed-effect covariates in `formula` are still used) within each level of a higher-level grouping variable (for example country, region, or survey wave) and reports how the variance partition coefficient (VPC/ICC) and the between-/within-stratum variance components differ across those groups. When the strata are defined by at least two dimensions it also fits the adjusted model (the dimensions' additive main effects) within each group and reports the per-group `pcv` – the proportional change in between-stratum variance, i.e. the additive share of that group's intersectional inequality.

Usage

```
compare_maihda_groups(
  formula,
  data,
  group,
  engine = "lme4",
  family = "gaussian",
  shared_strata = TRUE,
  min_group_n = 30,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  autobin = TRUE,
  decomposition = c("two-model", "crossed-dimensions"),
  context = NULL,
  sampling_weights = NULL,
```

```
  warn_linear = TRUE,
  estimation = c("fitted", "ML"),
  ...
)
```

Arguments

formula	A model formula. Either the shorthand intersectional form `outcome ~ covars + (1 | var1:var2)` (strata are built automatically) or `outcome ~ covars + (1 | stratum)` when `data` already contains a `stratum` column from `make_strata`. Any dimension main effect listed must be written as its bare column name: a transformed term such as `factor(edu)` is rejected, because only a bare name is recognised as a dimension's main effect and the transform would otherwise survive into the derived null model. Transform the column in `data` instead (`data$edu <- factor(data$edu)`).
data	A data frame containing the variables in `formula` and the grouping variable.
group	Character string naming the grouping variable in `data` (e.g. `"country"`). A separate model is fitted for each non-missing level.
engine	Modeling engine, "lme4" (default), "brms", "wemix" (the design-weighted fit; requires `sampling_weights` and is selected automatically when they are supplied with the default engine), or "ordinal" (cumulative link mixed model via `ordinal::clmm()`; selected automatically for an ordinal family or an ordered-factor outcome).
family	Model family. Default "gaussian". As in `fit_maihda`, a binary outcome is auto-detected once on the full data and switched to "binomial" (with a warning) so every group uses the same family.
shared_strata	Logical. When TRUE (default) intersectional strata are defined once on the full data so that a stratum denotes the same combination in every group; this makes the stratum *definitions* comparable across groups. Note that a group may still not contain every stratum, so two groups' VPCs can be estimated over different sets of populated strata – they are then not strictly directly comparable, and the function warns when this happens. When FALSE, strata are rebuilt independently within each group (stratum identities are then not comparable across groups at all).
min_group_n	Minimum size of the *analytic* sample a group must have – the rows that survive the model frame (covariate transformations applied, rows with a missing outcome/covariate dropped) – to be modelled. Groups with a smaller usable sample are skipped with a warning, even if they have more raw rows. Default 30.
bootstrap	Logical; compute per-group parametric-bootstrap VPC confidence intervals. lme4 engine only. Default FALSE.
n_boot	Number of bootstrap samples when `bootstrap = TRUE`. Default 1000.
conf_level	Confidence level for bootstrap intervals. Default 0.95.
autobin	Logical passed to `make_strata` controlling tertile binning of numeric grouping variables. Default TRUE.

`decomposition` Per-group additive-vs-interaction decomposition: the two-model null -> adjusted PCV (`"two-model"`, default) or the single crossed-dimensions model (`"crossed-dimensions"`). The crossed-dimensions form requires `shared_strata = TRUE` and at least two stratum dimensions, and adds the `var_additive`, `var_interaction`, `additive_share` and `interaction_share` columns (in place of `pcv` / `var_between_adjusted`); `var_between` is then the total between-strata variance (additive + interaction). See `maihda` for the underlying model and its caveats.

`context` Optional character vector naming higher-level *context* column(s) in `data` (e.g. `"school"`, `"region"`). Each per-group fit then becomes a contextual cross-classified model – `outcome ~ covars + (1 | stratum) + (1 | context)` – so within every group the stratum random intercept is crossed with the context random intercept(s). `vpc`/`var_between` are then the between-stratum quantities *net of* the context, and two columns report the per-group context partition: `var_context` (the between-context variance, summed over contexts) and `vpc_context` (the contexts' share of the group's unexplained variance). The per-context split is kept on the `"context_per"` attribute. A per-group subset shrinks each context's level count, so groups with too few context levels (< 10) to identify the context variance are named in a single warning. Forwarded to `fit_maihda`; not available for the `wemix`/`ordinal` engines (they fit no crossed random effect), and the `context` may not name the `group` variable itself.

`sampling_weights` Optional name of a sampling-weight column in `data` for design-weighted per-group fits; see `fit_maihda`. The column is sliced with each group's rows, so every group is fitted with its own members' weights. Not compatible with `engine = "lme4"`, `bootstrap = TRUE`, or (under the wemix engine) `decomposition = "crossed-dimensions"`.

`warn_linear` Logical; when `TRUE` (default) a one-time warning is emitted if a numeric stratum-defining dimension enters the per-group *adjusted* models as a raw linear slope rather than as categorical main effects, which silently changes the PCV/decomposition interpretation (see `maihda_adjusted_terms`). Set to `FALSE` internally when `maihda` delegates here, because it has already warned on the same data.

`estimation` Variance-estimation basis for each group's per-group PCV, `"fitted"` (default) or `"ML"`; see `calculate_pcv`. Affects Gaussian `lmer` fits only. Recorded as an `"estimation"` attribute on the returned comparison; the basis actually used is recorded as `"estimation_used"` (`"mixed"` when a group's ML refit was skipped at the boundary, leaving it on REML). Both are shown by `print()`. An `"ml_variances"` attribute is `TRUE` when the groups' fits estimate their variances by maximum likelihood (`glmer`, wemix or ordinal fits, or `lmer` fits made with `REML = FALSE`) or the PCVs compare maximum-likelihood refits (`estimation = "ML"`); under the default `estimation` `print()` notes it below a table with PCVs, as `calculate_pcv` does.

`...` Additional arguments passed to `fit_maihda` (and on to `lmer`/`glmer`). `weights`, `subset` and `offset` are an error on the `wemix`, `brms` and `ordinal` engines.

Details

It estimates one VPC per group as a stratified analysis: each group is modelled independently – the grouping variable itself is not modelled as a crossed random effect, and the strata are not adjusted

for it. (A `context` argument, if supplied, does add a crossed contextual random intercept *within* each group's model, making each per-group fit contextual cross-classified; the groups themselves remain independently fitted.)

The VPC is the *share* of the unexplained variance that lies between strata, not the absolute magnitude of intersectional inequality. Because it is a ratio, a group's VPC can differ from another's because the between-stratum variance differs, because the within-stratum (residual) variance differs, or both – two groups with the same between-stratum variance can have very different VPCs. To compare the absolute amount of between-stratum (intersectional) variation across groups, read the returned `var_between` column alongside the VPC rather than treating a higher VPC as "more inequality".

It is **descriptive**: it reports each group's VPC (with an interval when available – an lme4 bootstrap CI or a brms credible interval) for side-by-side comparison, but does not test whether the VPCs differ between groups. The per-group intervals describe each group's own uncertainty; whether two intervals overlap is *not* a valid test of the difference between their VPCs, which would require modelling that difference directly.

Robustness: a group whose *analytic* sample (rows surviving the model frame) has fewer than `min_group_n` observations is always skipped with a warning. A group with fewer than two populated strata is also skipped (VPC is undefined with a single stratum) when the stratum membership is known before fitting – that is, when `shared_strata = TRUE` or `data` already carries a `stratum` column. Under `shared_strata = FALSE` strata are rebuilt inside each group, so a degenerate single-stratum group is instead reported with a "fit failed" status rather than a pre-fit skip. A singular fit yields a VPC of 0 rather than an error (unlike `calculate_pcv`). A hard fit failure in one group records `NA` and a status note without aborting the whole comparison; one warning names the groups whose fit failed and gives each distinct reason once.

Fit-quality diagnostics: on every engine, one warning names the groups with a fit problem, and such a group keeps a `status` of `"ok"` (the fit did complete). A group is named for a singular fit when its between-stratum variance is estimated at zero, so that `var_between` and the VPC are 0. A zero in another variance component – a context, a further random effect, the interaction or a dimension of the crossed-dimensions model – leaves them as estimated and does not name the group. A group is also named when its model did not converge and, in a two-model decomposition, when its adjusted model did not. The warning says which: an adjusted model affects the group's PCV, not its VPC, and `pcv_status` does not record it. A `brms` model is named when its largest R-hat is above 1.01 or its sampler had a divergent transition, and is never reported as singular.

Value

A `data.frame` of class `maihda_group_comparison` with one row per group and columns `group`, `n`, `n_strata`, `vpc`, `var_between`, `var_other`, `var_residual`, `status` (and `ci_lower`/`ci_upper` whenever a group's summary supplies an interval: an lme4 bootstrap CI with `bootstrap = TRUE`, or a brms posterior credible interval, which is returned without bootstrapping). When the strata are defined by at least two dimensions, two further columns report the per-group null -> adjusted decomposition: `pcv` (proportional change in between-stratum variance when the dimensions' additive main effects are added; computed on the variance basis set by `estimation` – see `calculate_pcv`), `var_between_adjusted` (a *derived* coherence quantity, reported as `var_between * (1 - pcv)` so it shares the scale of the REML `var_between`/`vpc` and the table satisfies `pcv = (var_between - var_between_adjusted) / var_between` exactly – it is **not** the adjusted fit's own variance), and `var_between_adjusted_ml` (the adjusted model's *actual* between-stratum variance, read straight

off the adjusted fit on the same basis as the PCV; under the default `estimation = "fitted"` this is the REML variance, coinciding with `var_between_adjusted`, and under `estimation = "ML"` it is the ML variance, differing from `var_between_adjusted` only by the small REML-vs-ML gap in the null variance; despite its name it is not always a maximum-likelihood estimate; for a `brms` fit it is the posterior median of the adjusted variance, which `var_between_adjusted` equals as well). All three are `NA` for a group whose adjusted fit failed. A fourth column, `pcv_status`, records the decomposition outcome per group: `"ok"` when the PCV was computed, `"failed"` when the adjusted model or PCV errored (`pcv` is then `NA` and the group is named in a warning – the group's own `status` can still be `"ok"` because its null VPC model succeeded), and `"singular"` when the adjusted fit was singular and the PCV saturated near 100% – *not* warned about, since a singular adjusted fit is indistinguishable from genuinely additive strata (a legitimate, common result), so it is flagged only in this column. These four columns are omitted entirely when the strata have a single dimension. When `context` is supplied, two further columns report each group's contextual partition: `var_context` (the between-context variance, summed over contexts) and `vpc_context` (the contexts' share of the group's unexplained variance); the per-context split is on the `"context_per"` attribute and the context name(s) on `"context_var"`. These are dropped when no context is supplied. `n` is the analytic sample size used by the model (after dropping rows with a missing outcome/covariate) for both fitted and skipped groups, falling back to the raw row count only when the model frame cannot be built. `var_other` is the variance of any additional random effects and is 0 for the canonical single-stratum model. Groups that were skipped or failed have `NA` metrics and an explanatory `status`; for a failed fit it is `"fit failed: "` followed by the first line of the error, whose whole message is in a warning.

See Also

`compare_maihda` for comparing different models on the same data; `plot.maihda_group_comparison` for visualising the result.

Examples

```
data(maihda_country_data)
# How does gender x SES inequality in PISA math scores differ across countries?
cmp <- compare_maihda_groups(
  math ~ 1 + (1 | gender:ses),
  data = maihda_country_data,
  group = "country"
)
print(cmp)
plot(cmp, type = "vpc")
```

`fit_maihda` *Fit MAIHDA Model*

Description

Fits a multilevel model for MAIHDA (Multilevel Analysis of Individual Heterogeneity and Discriminatory Accuracy) using lme4, brms, WeMix (for design-weighted (survey) data), or – for an ordered-factor outcome – a cumulative link mixed model via `ordinal::clmm()`.

Usage

```
fit_maihda(
  formula,
  data,
  engine = "lme4",
  family = "gaussian",
  autobin = TRUE,
  context = NULL,
  sampling_weights = NULL,
  id = NULL,
  time = NULL,
  time_degree = 1,
  stratum_slope = TRUE,
  interactions = FALSE,
  count_approximation = c("lognormal", "delta", "trigamma"),
  ...
)
```

Arguments

`formula` A formula specifying the model. Can include a random effect for stratum (e.g., `outcome ~ fixed_vars + (1 | stratum)`) or can directly specify the intersection variables to be used for forming strata (e.g., `outcome ~ fixed_vars + (1 | var1:var2:var3)`). If variables other than "stratum" are provided in the random effect, `make_strata` will be called internally to compute the strata and the formula will be updated. A random slope, such as `(1 + x | stratum)`, is fitted, but `summary()`, the VPC, the PCV and the other MAIHDA summaries need intercept-only random effects outside a longitudinal model (see `id`). Fitted by `glmer()` (`glmer.nb()` for `family = "negbinomial"`) with `nAGQ` of 1 or more, a model with a random slope is refitted once when its fit stops with the variance of a block's first term, usually its intercept, at zero (in a block of three or more terms, also what is left of a later term's variance after the terms before it): glmer's first stage can leave it there, where the likelihood can still rise as that variance and the term's covariances with the later terms move off zero, and its second stage stops there too. The refit starts from the other side of that bound: with one later term, the sign of their covariance reversed; with two or more, along the direction in which the likelihood rises fastest. It is made by glmer's second stage with the `bobyqa` optimiser and replaces the fit when its -2 log-likelihood is lower by more than 1e-4. Such a model is then searched with each random-effect block's terms re-expressed as described for `lmer()` below and the fixed effects as the coefficients of centred and scaled columns, from lme4's starting values of the variance parameters with the fit's fixed effects and

from the fit's own estimates, and refitted from the lower end by glmer's second stage with bobyqa when its -2 log-likelihood is lower by more than 1e-4: with a slope covariate far from zero or in large units, such as a birth or calendar year, glmer's own second stage cannot resolve the slope's variance and can stop short of the maximum, often with what is left of that variance at zero, where lme4 calls the fit singular. Fitting then takes about one and a half times as long, and up to about four times. A fit refitted either way is refitted once more from its estimates by glmer's second stage with the optimiser your control names for that stage (`Nelder_Mead` by default; where Nelder-Mead's own first steps fail, or it stops without converging, its first steps are 0.1% of each estimate, at least 1e-6). This adds up to about three quarters to the time of a `glmer()` fit so refitted, and about doubles that of a `glmer.nb()` fit. Kept unless it ends higher than the refit or its optimiser reports a failure, the fit then records that optimiser, as glmer's own fits do; `lme4::refit()` of a fit without a control runs the optimiser the fit records, at its package's defaults. Fitted by `lmer()` outside a longitudinal model, a model with a random slope is refitted from the other side of such a bound when lmer's fit stops at one, and from a start carried back from the same model with each random-effect block's terms, other than an intercept, centred at their means and divided by their standard deviations in a block with an intercept, and divided by their root mean squares in a block without one, such as `(0 + x | ctx)` or the slope of `(1 + x || stratum)`. Besides stopping at the bound, lmer can stop short of the optimum when a slope covariate is far from zero or in large units, such as age in years or a calendar year, and where what is left of the variance of a block's last term after the terms before it is zero while the criterion still falls as that variance moves off zero; the fit kept and lmer's own fit are then refitted once more from that variance moved to the lowest criterion along it. A refit replaces lmer's fit when its REML criterion (deviance for a maximum-likelihood fit) is lower by more than 1e-4, the lowest of them kept; this takes about four times as long as one fit, and up to about nine times where lmer's own fit stops with a variance at zero. Pass `start` through `...` to fit from your own starting values only.

`data` A data frame containing the variables in the formula.

`engine` Character string specifying which engine to use: "lme4" (default), "brms", "wemix" (design-weighted pseudo-maximum-likelihood via `WeMix::mix()`; requires `sampling_weights`), or "ordinal" (cumulative link mixed model via `ordinal::clmm()`; requires an ordinal family). When `sampling_weights` is supplied and `engine` is left at its default, the engine switches to "wemix" automatically (with a message); likewise an ordinal family (or an auto-detected ordered-factor outcome) switches the default engine to "ordinal".

`family` Character string, family object, or family function specifying the model family. Common options: "gaussian", "binomial", "poisson", "negbinomial". Default is "gaussian". `family = "negbinomial"` fits an overdispersed count model with the dispersion parameter theta *estimated* from the data: lme4 via `lme4::glmer.nb()` and brms via its shape parameter (log link only; not supported by the wemix engine). A fixed-theta `MASS::negative.binomial(theta)` family object is also accepted with `engine = "lme4"` and is fitted with `glmer()`, honouring the supplied theta. With lme4 an estimated theta needs `nAGQ` of 0 or 1: `glmer.nb()` mis-estimates theta under adaptive quadrature, so `nAGQ > 1` is an error there

(a fixed theta is not affected). `family = "ordinal"` (alias `"cumulative"`; or `maihda_cumulative("probit")` / `brms::cumulative()` for a non-logit link) fits a cumulative (proportional-odds) model for an *ordered-factor* outcome: `ordinal::clmm()` under the automatic "ordinal" engine, `brms::cumulative()` under `engine = "brms"`. The VPC/ICC lives on the latent scale (level-1 variance $\pi^2/3$ logit / 1 probit, as for binomial models) and response-scale predictions are *expected category scores* (categories scored 1..K in order). An ordered-factor outcome with 3+ levels under the default family selects this model automatically, with a warning. The logit and probit links are supported; `sampling_weights`, `context`, and lme4-style `weights`/`subset`/`offset` arguments are not available on the clmm path. If the outcome variable appears to be binary and the default family is used, the function will automatically switch to "binomial", recode two-level responses to 0/1 for `glmer()`, and issue a warning. When a two-level non-0/1 response is recoded (on either the auto-detected or an explicit `family = "binomial"` path), the mapping follows the usual convention – the first level becomes 0 (reference) and the second becomes 1 (the modeled event), where "first/second" means alphabetical order for a character outcome and the declared order for a factor. The chosen mapping is reported via a `message()` and stored on the result as `$response_recoding`; set the factor levels (or supply a 0/1 outcome) to control which level is the event. Although any valid family object is accepted for fitting, the MAIHDA variance summaries (`summary.maihda_model`, VPC/ICC, PCV) are only defined for `gaussian("identity")`, the binomial/Bernoulli families with a logit, probit, or complementary log-log (`cloglog`) link (latent level-1 variance $\pi^2/3$, 1, and $\pi^2/6$ respectively), `poisson("log")`, and the negative binomial with a log link (level-1 variance `log(1 + 1/lambda + 1/theta)` at the *marginal* expected count `lambda`; Nakagawa, Johnson & Schielzeth 2017 – see `summary.maihda_model` for how `lambda` is computed). Other families (for example `Gamma(link = "log")`) will fit, but `summary()` and the VPC/PCV helpers will stop with an "not implemented" error because no level-1 variance is defined for them.

`autobin` Logical indicating whether numeric variables used only for automatic strata creation should be binned by `make_strata`. Default is TRUE.

`context` Optional character vector naming one or more higher-level *context* columns in `data` (e.g. `"school"`, `"hospital"`, `"region"`). Each enters the model as a crossed intercept-only random effect alongside the intersectional stratum effect – `outcome ~ covars + (1 | stratum) + (1 | context)` – giving the *contextual cross-classified MAIHDA* of the literature (individuals cross-classified by stratum and place/institution). `summary.maihda_model` then partitions the unexplained variance into between-stratum vs. between-context vs. residual, and the headline VPC/ICC remains the between-stratum share (now net of the context). A context variable may not be a stratum dimension or `"stratum"` itself, and may not already appear as a fixed-effect term (its variance would then be absorbed by the fixed part). A context with few levels (say < 10) identifies its variance weakly with either engine: `lme4` estimates it imprecisely, at zero (a singular fit) when the levels differ little, and with `engine = "brms"` its posterior depends on the prior on its standard deviation (under the default prior it comes out high when the context effects are smaller than that prior's scale, most with two or three levels), and the context share and the VPC depend on it. Prefer

contexts with many levels; with few, report the context share with its interval (a `brms` summary also gives the realised partition, which depends little on that prior when each context and stratum has enough data; see the section Realised strata and contexts of `maihda`), or enter the context as a fixed effect instead (`outcome ~ context + (1 | ...)`, without `context =`) when only the strata net of it are wanted. Writing the random effect directly in the formula (`... + (1 | school)`) fits the same model but is summarised generically as "Other random effects"; only `context =` activates the labelled contextual partition. Not supported by the `wemix` engine.

`sampling_weights`

Optional single character string naming a numeric column of `data` holding individual *sampling* (survey/design) weights, for a **weighted MAIHDA** on survey data. Sampling weights are not the same thing as lme4's `weights=` (precision weights, which rescale the residual variance), so combining `sampling_weights` with `engine = "lme4"` is an error.

What this does and does not buy you. A single person-level weight column is all this argument accepts: there is no representation of the sampling hierarchy – no primary sampling units, no sampling strata, no higher-stage or conditional weights, no finite-population corrections, and no replicate weights. Weighting the likelihood makes the point estimates target the population-weighted estimand, which is the main reason to use it. The standard errors, however, are *not* general design-based standard errors for a multistage or stratified sample, because none of the design features that drive them are supplied. Treat the intervals as approximate unless your design genuinely matches the assumptions below; for full design-based inference on such data, use replicate weights with a survey-specific package.

Two engines support them:

- `engine = "wemix"` (chosen automatically when `engine` is left at its default): weighted pseudo-maximum-likelihood via `WeMix::mix()` (Rabe-Hesketh & Skrondal 2006), the estimator used for NAEP/PISA analysis. The individual weights enter at level 1 unchanged and the level-2 (stratum) weights are 1, because intersectional strata are treated as exhaustive population cells included with certainty – so this is a *single-stage* weighted model, and it is that assumption the inference rests on. Supports `gaussian(identity)` and `binomial(logit)` models with the canonical single `(1 | stratum)` random intercept and no `offset()` term: `WeMix::mix()` leaves an offset out of the model it fits, so a formula with one is an error. For a Gaussian outcome, fit the response minus the offset instead (the same model) and add the offset back to its predictions. A `binomial(logit)` model whose between-stratum variance is estimated at zero is an error as well: `WeMix::mix()` cannot return that fit. It is a common result for the adjusted model when the strata are additive, and `engine = "brms"` returns a fit there. Fixed-effect standard errors are the sandwich (robust) errors WeMix reports, which account for the weighting and for dependence within the model's own grouping (the intersectional strata) – but not for clustering or stratification induced by the sample design, which the strata do not represent. The VPC/PCV are reported as point estimates (no bootstrap – see `summary.maihda_model`).

- `engine = "brms"`: the weights enter the model as likelihood weights (`y | weights(w)`), normalized to mean 1, giving a *pseudo-posterior*: point estimates target the population-weighted estimand but credible intervals are not design-based – interpret them cautiously. Full design-based (replicate-weight / linearised) variances for the variance components are not computed.

Rows with a missing or non-positive sampling weight are dropped with a warning. The column names `.maihda_sw` (brms likelihood weight) and `.maihda_l2wt` (WeMix level-2 weight) are **reserved** for the design-weighted engines: they are written into the analytic data internally, so a model that supplies `sampling_weights` may not also reference a variable of either name in its formula (fitting would error). Default `NULL` (unweighted).

`id` Optional single character string naming a person/unit identifier column for a **longitudinal (growth-curve) MAIHDA** on long-format data (one row per measurement occasion). Id values must be *globally* unique to a person – ids numbered within a site or group (person "1" in every site) would merge different people's trajectories, and an id appearing in more than one stratum is rejected with an error. Supplied together with `time`, it makes the model a 3-level growth curve – occasions within individuals (`id`) within intersectional strata – with a random intercept and slope on `time` at *both* the individual and stratum levels. The growth random effects are added automatically: write the strata shorthand (`(1 | var1:var2)` (or `(1 | stratum)`) only, not the slopes. The between-stratum variance (and hence the VPC) then becomes a function of time; `summary.maihda_model` reports the time-varying VPC. Longitudinal fits are supported by `engine = "lme4"/"brms"` only (not `wemix`/`ordinal`), and are incompatible with `context` and `sampling_weights`. Default `NULL` (cross-sectional). See Bell, Evans, Holman & Leckie (2024).

`time` Optional single character string naming a numeric measurement-time column (e.g. wave 0, 1, 2, ... or age), required for a longitudinal MAIHDA; see `id`. When the time axis does not start at 0 (age, calendar year, waves coded 10, 11, ...), the growth terms are fit on internally *centered* time (`time - min(time)`, with a message): the raw polynomial basis over an offset range is ill-conditioned and can silently converge to a wrong solution. With `engine = "lme4"` the model is also fit on time divided by the power of ten that puts its span between 1 and 10 (days over five years are divided by 1000; a span already between 1 and 10 is left as it is), with a message: counted in fine units (days, or months for a quadratic curve), lme4's optimisers stop far from the optimum, mostly without a convergence warning. A model fitted by `glmer()` (`glmer.nb()` for `family = "negbinomial"`) is refitted as well: glmer's first stage often leaves a growth block's intercept variance at zero, where the likelihood still rises with the sign of the intercept-slope covariance reversed, and its second stage stops there. Linear growth with a stratum slope is refitted from two more starts, which reset the person-level block and start the stratum block at half the person-level intercept SD and 0.71 times its slope SD, with an intercept-slope correlation of 0.71 or -0.71; a refit replaces the fit when its -2 log-likelihood is lower by more than 1e-4. Then any glmer growth fit that still has a block's intercept variance at zero is refitted from the other side of that bound by glmer's second stage with the `bobyqa` optimiser and, when that refit is kept, once more: across the bound

again if it stops at one, otherwise from where it stops; each refit is kept on the same terms. Every glmer growth fit is then searched as a glmer model with a random slope is (see `formula`), with the penalised least squares inside each likelihood evaluation run to a relative tolerance of 1e-10 (lme4's is 1e-7), and refitted from the lower end when its -2 log-likelihood is lower by 0.001 or more: glmer's own second stage stops quadratic growth fits short of the maximum, often near a zero variance of the quadratic term, where lme4 calls the fit singular. This takes about three times as long as one fit; the searches add about a fifth to that, and a refit from them about as much again. A fit any of these refits replaced is then refitted once more from its estimates by glmer's second stage with the optimiser your control names for that stage, as described for `formula`, which adds about a twentieth. `lme4::refit()` of a glmer growth fit on a simulated response (`lme4::bootMer()` makes one for each draw) runs the optimiser the fit records from the fit's estimates, and `Nelder_Mead`, the default, often stops short of the response's maximum there; the bootstraps of `summary()` refit each draw as `fit_maihda()` fits the model. Pass `start` through `...` to fit from your own starting values only; they are on the internal time when `time` is scaled. All results (the time-varying VPC, the PCV, the fixed effects in `summary()`, plots, predictions) are reported on the original `time` scale; the lme4 fit stored in the result's `model` element is on the internal time. The column name `.maihda_ctime` is reserved for the internal time variable. Default `NULL`.

`time_degree` Polynomial degree of the growth curve when `time` is supplied: 1 (default) linear, 2 quadratic, etc. The brms engine supports degree 1 only.

`stratum_slope` Longitudinal only: keep the stratum-level random slope(s) on `time`? `TRUE` (default) fits the canonical Bell et al. (2024) structure, `(time | id) + (time | stratum)`, in which the between-stratum variance is a function of time. `FALSE` fits `(time | id) + (1 | stratum)`: the individual level keeps its growth block, but strata differ in *level* only, so the between-stratum variance – and the numerator of the VPC – is constant over time and no `PCV_slope` is defined. It assumes that the strata do not diverge over time. A `(time | stratum)` block that collapses to a perfect intercept-slope correlation does not show that: few strata, few occasions per stratum or irregular measurement times produce such fits when the strata diverge too, and where they do, `FALSE` moves their slope differences into the person-level slope variance. The VPC still varies with time through the person-level slope variance and the residual, so this is a time-constant between-stratum *variance*, not a time-constant VPC. A Gaussian growth model fitted by `lmer()` with a stratum slope, or with quadratic or higher growth, is fitted from lme4's starting values and again from more starts, and a refit replaces lmer's fit when its REML criterion (deviance for a maximum-likelihood fit) is lower by more than 1e-4: lme4's default optimiser can stop at a boundary of a random-effect block away from the optimum. With linear growth the starts are scaled by the person-level block, with a positive and a negative stratum intercept-slope correlation; with quadratic or higher growth one start is lmer's fit of the same model on time centred at the middle of its range, and, when that refit and lmer's fit differ by more than 0.01, another has both covariance blocks diagonal. Pass `start` through `...` to fit from your own starting values only; they are on the internal time when `time` is scaled (see `time`).

`interactions` Opt-in per-stratum interaction diagnostic (`maihda_interactions`), attached as

the interactions slot and shown by print(). FALSE (default) skips it; TRUE computes it with the diagnostic's default correction (adjust = "BH"); or pass a p.adjust method name, including "none" for the uncorrected view. It is meaningful only on an *adjusted* model (the dimensions' main effects in the fixed part); on a null model maihda_interactions warns. This is the single-fit parallel to the default-on interactions of maihda.

count_approximation

Which latent-scale level-1 (observation) variance approximation the VPC/ICC of a *log-link count* model uses, from table 1 of Nakagawa, Johnson & Schielzeth (2017): "lognormal" (default, $\ln(1 + 1/\lambda\ [+\ 1/\theta])$), "delta" ($1/\lambda\ [+\ 1/\theta]$), or "trigamma" ($\psi_1(\lambda)$ for Poisson, $\psi_1((1/\lambda + 1/\theta)^{-1})$ for the negative binomial), with λ the marginal expected count (see summary.maihda_model). The three agree above about $\lambda = 2$ and diverge sharply below it, where the VPC depends heavily on the choice; summary() reports the method and λ, and warns below that threshold. Nakagawa et al. recommend the trigamma form; at low counts it gives the smallest VPC of the three. The default is the approximation performance::icc() uses, but evaluated at a different λ on an adjusted model, so the two VPCs agree exactly only for a null model. Used by summary(), the bootstrap intervals and the longitudinal VPC(t); ignored for every other family. The count-scale VPC of Leckie et al. (2020), from maihda_vpc_response, partitions the variance of the counts themselves and does not depend on this choice.

...

Additional arguments passed to lmer/glmer (lme4), brm (brms), or WeMix::mix() (wemix; e.g. nQuad, fast). The wemix engine rejects mix()'s center_grand and center_group, under any partial spelling: WeMix centres the covariates internally without keeping the centring constants, so predictions could not reproduce the fit. Centre covariates in data before fitting instead. It likewise rejects an offset() term in the formula (see sampling_weights). The lme4-style weights (precision weights), subset, and offset arguments are honoured only by the lme4 engine, which applies them directly. The wemix, ordinal, and brms engines reject them: none takes them as a top-level fitting argument (brms in particular expects weighting/offset as formula addition terms, weights(.) / offset(.), and design weights via sampling_weights). Prefilter data instead of using subset on those engines. On engine = "brms" the response may carry the trials() and weights() addition terms (y | trials(n), y | weights(w)); any other – se(), rate(), cens(), trunc(), mi(), ... – is refused, since the VPC, predictions and summaries do not model it. Write an exposure as + offset(log(expo)) rather than rate(expo): for a Poisson model the same model, which they do (for a negative binomial, brms's rate() also scales the shape by the exposure, which the offset does not). An engine = "brms" fit without a seed takes its sampler seed from R's random-number stream, so set.seed() before the call reproduces it, with the chains run in parallel (cores > 1) as well.

Value

A maihda_model object containing:

model

The fitted model object (lme4, brms, WeMix, or ordinal::clmm). An ordinal fit with a between-stratum variance of zero comes from clmm()'s general rou-

tine, `clmm.control(useMatrix = TRUE)`, the one that can return it. An ordinal model with fixed effects is fitted on its design with the columns centred and scaled (scaled only under `threshold = "symmetric2"`), since a covariate far from zero or on a large or tiny scale can stall the optimiser, and is returned in the formula's own terms with the gradient and Hessian of that fit, which give its standard errors; a fit given a `start` is made as the formula stands. An lme4 fit's call gives its control as a call to `lme4::lmerControl()` or `lme4::glmerControl()`, so `lme4::bootMer()` and `confint(method = "boot")` refit it as they refit lme4's own fits. A glmer fit that `fit_maihda()` last refitted by bobyqa, or by Nelder-Mead with first steps of its own where Nelder-Mead's own fail, names that refit's control instead, bound in the formula's environment, and both stop on it: a draw refitted from the fit with those steps can stop short of its maximum. A control whose optimiser is a function is named the same way, as `lme4::refit()` does not take one

`engine` The engine used ("lme4", "brms", "wemix", or "ordinal")

`sampling_weights`
The sampling-weight column name when supplied, NULL otherwise

`formula` The model formula

`data` The data used for fitting

`family` The family used

`strata_info` The strata information from make_strata() if available, NULL otherwise

`context_vars` The context variable name(s) when `context` was supplied, NULL otherwise

`interactions` The `maihda_interactions` diagnostic when `interactions` is not `FALSE`, NULL otherwise

`response_recoding`
For a recoded two-level outcome, a data frame mapping each original level to its 0/1 value and role (reference/event); NULL when no recoding occurred

`diagnostics` Fit-quality diagnostics, surfaced by the print and summary methods: singular fit / convergence for lme4, WeMix and ordinal (clmm) fits, and MCMC convergence (maximum Rhat, divergent transitions) for brms. lme4 checks a fit in the units of its covariates (a `glmer()` fit in those of its fixed effects and random slopes, an `lmer()` fit in those of its random slopes), so its checks can fire at the maximum when such a covariate is far from zero or in large units, as a calendar year or an income is. When they fire and the optimiser reports success, the fit counts as converged if it lies within a tenth of a standard error of a stationary point of its likelihood (the restricted likelihood for an lmer fit by REML), measured with those covariates centred and scaled (a slope in a `diag()` block only scaled); the distance is `stationary_distance`, and lme4's messages are kept in `lme4_overruled` when the fit converged by it. lme4's singular-fit test, a diagonal entry of a random-effect block's relative Cholesky factor below 1e-4, is taken in the same units and can fire on a block with a slope on such a covariate that is not at the boundary: in an lmer fit, or a glmer fit with `nAGQ` of 1 or more, outside a longitudinal model, a block with a slope is at the boundary when a diagonal entry of its factor with the terms centred and scaled (in a block without an intercept, divided by their root mean square) is below 0.01, and lme4's own verdict is `lme4_singular` where the two differ. A fit with a

diag(), cs() or ar1() block keeps lme4's test. A glmer fit (nAGQ of 1 or more), or an lmer fit outside a longitudinal model, with the variance of a random-effect block's first term, usually its intercept, at zero (or, in a block of three or more terms, what is left of a later term's variance after the terms before it, such as a quadratic growth block's slope perfectly correlated with its intercept), where the likelihood still rises as the block's variances and covariances move, or as that variance and the term's covariances with the later terms move off zero, has stopped at a bound of lme4's parametrisation and counts as not converged; those blocks are in boundary_traps. A slope's entry can lie within lme4's 1e-4 of that bound at its natural size when its covariate is in large units, as a covariate in thousands is; such a column is not at a bound (at_bound is FALSE), and the fit counts as not converged when the likelihood still rises as the block's variances and covariances move. An lmer fit outside a longitudinal model that lme4 calls singular, which lme4 does not check further, or one with a random slope that lme4 does not check because of its size (by default one of 10,000 rows or more, or with 10 or more variance parameters), counts as not converged when a search of the same model, from lme4's starting values with each random-effect block's terms centred and scaled and from a variance at zero moved off it, reaches a REML criterion (deviance for a maximum-likelihood fit) 0.01 or more below the fit's; that amount is search_gain. A fit refitted as described under formula has been through those searches already, so this concerns a fit made from your own start or a singular one without a random slope. A glmer fit with nAGQ of 1 or more and a random slope, outside a longitudinal model and with no diag(), cs() or ar1() block, that lme4 calls singular is searched from the fit with its random-effect terms and fixed effects centred and scaled, and counts as not converged when the search reaches a deviance 0.01 or more below the fit's (search_gain); such a fit refitted as described under formula has been through such searches already and is not searched again. A glmer fit with nAGQ of 1 or more and a random slope, outside a longitudinal model, that lme4 does not check because of its size (by default one of 10,000 rows or more, or with 20 or more parameters, fixed effects included) and does not call singular is measured as one whose checks fired, and counts as not converged, with lme4_unchecked TRUE, a tenth of a standard error or more from a stationary point; where that cannot be measured it is searched in the same way. An lme4 fit also carries likelihood-adequacy caveats – count overdispersion and zero inflation, stratum random-effect non-normality and longitudinal residual autocorrelation – reported only when a conservative threshold is crossed, since the VPC/PCV and interaction estimates are conditional on the likelihood holding. Every one of those checks is lme4-only, so a cumulative (clmm) fit raises no adequacy caveat at all: it stores a single descriptive fixed-only proportional-odds proxy statistic, never flagged because it cannot be separated from stratum heterogeneity; use maihda_proportional_odds_test to test that assumption

Examples

```
# Standard approach: manually create strata first
strata_result <- make_strata(maihda_sim_data, vars = c("gender", "race", "education"))
model <- fit_maihda(health_outcome ~ age + (1 | stratum),
                    data = strata_result$data,
```

```
                    engine = "lme4")

# Simplified approach: specify stratifying variables directly in the grouping structure
# The function internally calls make_strata() to create intersectionals
model2 <- fit_maihda(health_outcome ~ age + (1 | gender:race:education),
                     data = maihda_sim_data,
                     engine = "lme4")

# Contextual cross-classified MAIHDA: strata crossed with a higher-level
# context (here country) -- the literature's cross-classified MAIHDA.
data(maihda_country_data)
model3 <- fit_maihda(math ~ 1 + (1 | gender:ses),
                     data = maihda_country_data,
                     context = "country")
summary(model3)  # between-stratum vs. between-country vs. residual
```

`maihda` *Run a Complete MAIHDA Analysis*

Description

A single high-level entry point that runs the standard two-model MAIHDA workflow and returns one bundled object. It fits the **null** model (covariates plus the intersectional random intercept, *excluding* the stratum dimensions' main effects) and the **adjusted** model (the null plus the additive main effects of the stratum-defining dimensions), summarises the variance partition (VPC/ICC) of the null model, and reports the **PCV** – the proportional change in between-stratum variance from the null to the adjusted model, i.e. the additive share of the intersectional inequality. When a higher-level grouping variable is supplied it also compares this decomposition across that variable's levels.

Usage

```
maihda(
  formula,
  data,
  group = NULL,
  context = NULL,
  engine = "lme4",
  family = "gaussian",
  decomposition = c("two-model", "crossed-dimensions", "longitudinal"),
  autobin = TRUE,
  shared_strata = TRUE,
  min_group_n = 30,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
```

```
  response_vpc = FALSE,
  seed = NULL,
  sampling_weights = NULL,
  id = NULL,
  time = NULL,
  time_degree = 1,
  stratum_slope = TRUE,
  interactions = TRUE,
  estimation = c("fitted", "ML", "REML"),
  ...
)
```

Arguments

formula	A model formula, using either the intersectional shorthand `outcome ~ covars + (1 | var1:var2)` or `... + (1 | stratum)` when `data` already has a `stratum` column from `make_strata`. The dimensions' additive main effects may be listed (the fully-specified adjusted model) or omitted (added automatically, with a message); see Details. Any dimension main effect listed must be written as its bare column name: a transformed term such as `factor(edu)` is rejected, because only a bare name is recognised as a dimension's main effect and the transform would otherwise survive into the derived null model. Transform the column in `data` instead (`data$edu <- factor(data$edu)`). For the same reason a fixed interaction involving a dimension is rejected, whether with another dimension (`gender * race`) or with a covariate (`age * gender`): only the bare main effect is removed, so the interaction survives into the null model, which would then already adjust for that dimension.
data	A data frame with the model variables (and the `group` variable, if used).
group	Optional character string naming a higher-level grouping variable (e.g. `"country"`). When supplied, `compare_maihda_groups` is run and attached to the result. `group` runs a *stratified* analysis (one independent model per level); to instead model a higher level *jointly*, crossed with the strata, use `context`. The two are different designs but they *compose*: supplying both runs the stratified comparison where each per-group fit is itself a contextual cross-classified model (see `context`).
context	Optional character vector naming higher-level *context* column(s) in `data` (e.g. `"school"`, `"hospital"`, `"region"`), forwarded to `fit_maihda`. Each context enters every fitted model as a crossed random intercept – `outcome ~ covars + (1 | stratum) + (1 | context)` – the *contextual cross-classified MAIHDA* of the literature. The summaries then partition the unexplained variance into between-stratum vs. between-context vs. residual, the headline VPC/ICC becomes the between-stratum share *net of* the context, and the PCV decomposition is computed with the context partialled out (the context random intercept is carried by both the null and the adjusted model). May be combined with `group`: the stratified comparison then fits a contextual cross-classified model within each group and reports each group's stratum-vs-context partition (a per-group subset shrinks the context level count, so weak identification is warned about per group). Not available for the `wemix`/`ordinal` engines (they fit no crossed random effect). A context with few levels identifies its variance weakly with either engine: `lme4`

estimates it imprecisely, at zero (a singular fit) when the levels differ little, and with `engine = "brms"` its posterior depends on the prior on its standard deviation – with two levels roughly in proportion to the prior's scale – and under the default prior it comes out high when the context effects are smaller than that prior's scale, most with two or three levels; the context share and the VPC depend on it. `print()` of the analysis and of its `summary()` names the contexts of a `brms` fit that have fewer than 10 levels, and the contextual partition of a `brms` summary holds the level counts as `context_levels`. A `brms` fit also reports the realised partition: the variances and shares of the effects of the strata and context levels in the data, which depend little on that prior when each context and stratum has enough data (see the section Realised strata and contexts). Prefer contexts with many levels (see `context` in `fit_maihda` for what to do with few).

`engine` Modeling engine, "lme4" (default), "brms", "wemix" (the design-weighted pseudo-maximum-likelihood fit; requires `sampling_weights` and is selected automatically when they are supplied with the default engine), or "ordinal" (cumulative link mixed model via `ordinal::clmm()`; selected automatically for an ordinal family or an ordered-factor outcome).

`family` Model family. Default "gaussian". As in `fit_maihda`, a binary outcome is auto-detected when `family` is left at the default, and the same resolved family is then used for the group comparison so all models agree.

`decomposition` How to decompose the intersectional inequality into additive and interaction parts. `"two-model"` (default) is the classic MAIHDA approach: a null model and an adjusted model (the dimensions' additive main effects as *fixed* effects), with the additive share read from the PCV (proportional change in between-stratum variance). `"crossed-dimensions"` fits a **single** model that enters each dimension's additive main effect as a *random* intercept – `outcome ~ covars + (1 | dim1) + ... + (1 | stratum)` – so each dimension's RE variance is its additive contribution and the intersection (`stratum`) RE variance is the interaction beyond additive; the additive and interaction *shares* of the total between-strata variance are read directly from that one fit. The two modes target the same scientific question with different estimators, so their additive shares are conceptually parallel but not numerically identical. The crossed-dimensions additive share is a partial-pooling estimate, and each dimension's variance is estimated from that dimension's levels alone, so a dimension with few levels (a binary sex variable has two) is poorly identified with either engine. `lme4` often returns a singular fit. With `engine = "brms"` the posterior of such a variance depends on the prior on its standard deviation – with two levels roughly in proportion to the prior's scale – and the VPC and the additive share follow it: under the default prior they are overstated, most with two- to four-level dimensions, and with two- or three-level dimensions the additive share comes out high even where the dimensions have no additive effect. `print()` names the dimensions of a `brms` fit that have fewer than 10 levels. A `brms` fit also reports the realised strata: the VPC, shares and variances of the effects of the strata and dimension levels in the data, which depend little on that prior when each stratum has enough data, whatever the number of levels (see the section Realised strata). Otherwise prefer `"two-model"` (with few strata see `estimation`) unless every dimension has about ten or more levels. See Details. (In the MAIHDA literature

"cross-classified" refers to the contextual stratum-by-place model, which this package fits via `context`.) `"longitudinal"` fits a 3-level **growth-curve** MAIHDA (requires `id` and `time`; selected automatically when they are supplied): a null and an adjusted growth model, where the adjusted model adds the dimensions' main effects *and their interactions with time* (`dim:time`). The between-stratum variance is then time-varying and the PCV is reported separately for the baseline (intercept) and the slope variance – the additive-vs-multiplicative split of the intersectional trajectory inequality (Bell, Evans, Holman & Leckie 2024). As with the two-model PCV, the `estimation` argument sets the variance-estimation basis: the default `"fitted"` keeps each REML `lmer` growth fit's own variance, while `"ML"` refits them with maximum likelihood before the null-vs-adjusted comparison, which overstates the PCV when strata are few (see `calculate_pcv`). The reported time-varying VPC always keeps each fit's own (REML) estimate. See `fit_maihda`.

`autobin` Logical passed to `make_strata`; tertile-bins numeric grouping variables. Default TRUE.

`shared_strata` Logical, forwarded to `compare_maihda_groups` when `group` is supplied: build strata once on the full data so VPCs are comparable across groups (TRUE, default) or rebuild them within each group.

`min_group_n` Minimum group size for the per-group comparison, forwarded to `compare_maihda_groups`. Default 30.

`bootstrap` Logical; compute parametric-bootstrap VPC confidence intervals (lme4 only) for both the overall summary and the per-group comparison. Default FALSE.

`n_boot` Number of bootstrap samples when `bootstrap = TRUE`.

`conf_level` Confidence level for bootstrap intervals. Default 0.95.

`response_vpc` Logical; for a binomial (lme4) outcome, also attach the response-scale VPC (`maihda_vpc_response`) to the model summaries. It is estimated by simulation, so it is opt-in (default `FALSE`) and uses `seed`. For a Poisson or negative-binomial (lme4) outcome it attaches the count-scale VPC instead, a closed form (see `summary.maihda_model`). The discriminatory accuracy (AUC + MOR) is attached automatically for a binomial/Bernoulli outcome regardless of this flag (see Details).

`seed` Optional integer seed for the response-scale VPC simulation. It is NOT forwarded to `brms::brm()`: an `engine = "brms"` fit's MCMC sampler is unseeded by it (a warning says so). Call `set.seed()` before `maihda()` to make a Bayesian fit reproducible, or use `fit_maihda(seed = )`, which has no `seed` formal and so passes it through to `brm()`.

`sampling_weights`
Optional name of a sampling-weight column for a **weighted MAIHDA** on survey data; see `fit_maihda` for the full semantics (engine selection, the pseudo-likelihood weighting, and – importantly – why a single person weight does not deliver design-based inference for a multistage or stratified sample). Both the null and the adjusted model (and any per-group fits) use the same weights, so the PCV is a weighted decomposition. Not compatible with `engine = "lme4"`, `decomposition = "crossed-dimensions"` under the wemix engine, or `bootstrap = TRUE`.

`id`, `time`, `time_degree`
: For a **longitudinal** MAIHDA: the person/unit identifier column, the numeric measurement-time column, and the growth-curve polynomial degree (1 = linear). Supplying `id`/`time` selects `decomposition = "longitudinal"`. See `fit_maihda` for the model structure; `group`, `context`, and `sampling_weights` are not supported alongside them. Default `NULL` (cross-sectional).

`stratum_slope`
: Longitudinal only: keep the stratum-level random slope(s) on `time`? `FALSE` fits a random intercept at the stratum level, so the between-stratum variance is constant over time and the decomposition reports `PCV_intercept` only. Applied to the null and adjusted growth models alike. See `fit_maihda`.

`interactions`
: Whether to compute the per-stratum interaction diagnostic (`maihda_interactions`) on the adjusted / crossed-dimensions model and attach it as the `interactions` slot, surfaced in `print()`. `TRUE` (default) uses the diagnostic's own default correction (`adjust = "BH"`, FDR – since scanning all strata is a screening question); `FALSE` skips it; or pass a `p.adjust` method name – including `"none"` for the uncorrected, per-stratum individual-testing view. Uses `conf_level`. Not computed for a longitudinal decomposition. The computation is cheap (it reads the stratum estimates the summary already holds; no refit).

`estimation`
: Variance-estimation basis for the PCV (`"two-model"` and `"longitudinal"` decompositions), `"fitted"` (default), `"ML"` or `"REML"`; see `calculate_pcv`. `"fitted"` differences each model's own between-stratum variance (the REML estimate for a Gaussian `lmer` fit, as `summary()` reports); `"ML"` refits REML `lmer` fits with maximum likelihood first. `"REML"` compares REML-type variances for a binomial or Poisson lme4 fit in the `"two-model"` decomposition, whose maximum-likelihood PCV runs high with few or small strata; it is an error with `group`, with `context` on such a fit, and in the `"longitudinal"` decomposition.

`...`
: Additional arguments passed to `fit_maihda` (and on to `lmer`/`glmer`). Includes `count_approximation`, which selects the log-link count level-1 variance approximation for the VPC and is applied to every derived fit.

Details

Binomial companions. For a binary outcome the model summaries also carry the discriminatory accuracy (AUC / C-statistic and Median Odds Ratio) – the "DA" in MAIHDA – automatically, so the null model's strata-only AUC sits alongside its VPC; set `response_vpc = TRUE` to add the (simulation-based) response-scale VPC as well. These are read from `summary(x)` and the attached `summary_adjusted`, and the headline `print()` shows the null-model AUC.

This is a convenience wrapper around `fit_maihda`, `calculate_pcv`, `summary.maihda_model` and `compare_maihda_groups`. It *always* runs a decomposition, never a single plain fit: by default the two-model null/adjusted pair, or – with `decomposition = "crossed-dimensions"` – one crossed model whose additive dimension random effects carry the additive share. For a single undecomposed fit (e.g. just the null-model VPC / discriminatory accuracy), call `fit_maihda` directly.

The dimensions' additive main effects. You may write them in the formula – the fully-specified, lme4-native adjusted model `outcome ~ covars + var1 + var2 + (1 | var1:var2)` – or omit them. Either way the null *excludes* the dimension main effects and the adjusted *includes* them: when the formula already lists them it is taken as the adjusted model and the null is derived by dropping them; when they are missing `maihda()` adds them to the adjusted model and emits a `message()`

so the decomposition stays explicit. Only the *additive* main effects belong here: a fixed interaction among the stratum dimensions – `var1 * var2`, which R expands to `var1 + var2 + var1:var2` – duplicates the intersectional stratum random intercept (it absorbs the between-stratum variance into fixed cell means, which makes the PCV invalid), so it is **rejected with an error**. Write `var1 + var2`; the intersection is estimated by the stratum random effect (and quantified by `decomposition = "crossed-dimensions"` or `maihda_interactions`). The dimensions themselves are read from the random term: the shorthand `(1 | var1:var2)` and `make_strata` both record them, and a numeric dimension that `make_strata()` auto-binned enters the adjusted model as its reconstructed tertile factor (matching the strata), not as a linear term. Because `maihda()` is intrinsically a decomposition, it **errors** (rather than returning a null-only result) when it cannot build the adjusted model – when the dimensions cannot be recovered (a hand-built `stratum` column records none) or there is only one dimension (no intersection to decompose). Use `fit_maihda` for those single-model fits.

Value

An object of class `maihda_analysis`: a list with

`model` the fitted `maihda_model` (see `fit_maihda`); the **null** model in `"two-model"` mode, or the single **crossed-dimensions** model in `"crossed-dimensions"` mode

`summary` the model's `maihda_summary` (VPC/ICC, variance components, stratum estimates; plus the additive/interaction `decomposition` in crossed-dimensions mode, and the stratum-vs-context `context` partition when `context` is supplied)

`model_adjusted` the fitted **adjusted** `maihda_model` (`"two-model"` and `"longitudinal"` modes; `NULL` otherwise)

`summary_adjusted` the adjusted model's `maihda_summary`, or `NULL`

`pcv` the proportional change in variance: the `pcv_result` from `calculate_pcv` in `"two-model"` mode, or a `maihda_long_pcv` (the intercept/slope and time-specific PCV) in `"longitudinal"` mode; `NULL` otherwise. Both use the variance-estimation basis set by `estimation`: the default `"fitted"` keeps each Gaussian `lmer` fit's own REML variance (matching the single-model summaries), while `"ML"` refits the REML fits with maximum likelihood, which overstates the PCV when strata are few (see `calculate_pcv`). In `"longitudinal"` mode both PCVs, the slope's first, are understated on average when the strata differ little after adjustment next to the precision of one stratum's own estimate. For `engine = "brms"` in `"longitudinal"` mode the variances compared are posterior medians of the per-draw between-stratum variance – at the baseline, of the slope, and at each reporting time – and, like the two-model `brms` PCV, the result is understated on average when strata are few

`decomposition` the additive/interaction partition (additive and interaction variances and shares, per-dimension variances and, for a `brms` fit, each dimension's number of levels in `dim_levels` and the realised strata in `realised`, described in the section Realised strata; `"crossed-dimensions"` mode only, `NULL` otherwise)

`groups` a `maihda_group_comparison` when `group` is supplied, otherwise `NULL`

`interactions` the `maihda_interactions` diagnostic (per-stratum interaction BLUPs and flags) when `interactions` is not `FALSE`, otherwise `NULL`

`mode` `"two-model"`, `"crossed-dimensions"`, or `"longitudinal"`

`context_vars` the context variable name(s) when `context` was supplied, otherwise `NULL`

`formula`, `adjusted_formula`, `group_var`, `call`
bookkeeping for printing

Numeric stratum dimensions

The stratum-defining dimensions are categorical. A factor or character dimension enters the adjusted model as a categorical main effect, and a numeric dimension with more than 10 unique values is tertile-binned when `autobin = TRUE` (see `make_strata`). A numeric dimension that is *not* auto-binned – a category code such as `education = 1, 2, 3` (few unique values), or a many-valued numeric fitted with `autobin = FALSE` – enters the adjusted model as a single *linear* fixed effect rather than categorical main effects, which changes the PCV/decomposition interpretation. `maihda()` therefore **warns** in that case (for three or more distinct values – a binary numeric dimension is exempt, since with two levels a linear term and a factor are equivalent): wrap category codes in `factor()` before fitting, or, for a genuinely continuous variable, use it as a covariate rather than a stratum dimension.

Realised strata

For a `brms` crossed-dimensions fit, `decomposition$realised` and the "Realised strata" block of `print()` summarise, over the posterior draws, the effects of the J strata in the data. In each draw stratum j has the effect ζ_j, the sum of its dimensions' level effects and its own (interaction) effect. These are finite-population summaries (Gelman 2005): they describe the strata and levels in the data, where the variance components describe the distributions those were drawn from. A dimension with two levels says a lot about the former and little about the latter, so the realised values depend little on the prior on the standard deviations when each stratum has enough data to pin down its own effect. With few observations per stratum each draw also carries the uncertainty about every stratum's effect: the realised variances then come out too high and move with the prior. Compare the intervals of the stratum estimates with the spread between the strata. Per draw:

`between_var` $\sum_j(\zeta_j - \bar{\zeta})^2/(J-1)$, half the mean squared difference between two distinct strata.

`vpc` `between_var` over itself plus the within-stratum variance: $\pi^2/3$ for a logit link, 1 for a probit link, $\pi^2/6$ for a cloglog link, the residual variance of a Gaussian fit. Not given for a Poisson or negative-binomial fit, whose within-stratum variance is computed from the variance components. With a `context`, the realised variance of the context effects is added to the denominator (see the section Realised strata and contexts).

`interaction_var` the residual sum of squares of the least-squares fit of the ζ_j on an intercept and the dimensions' levels, divided by its residual degrees of freedom, `df_interaction`. Strata missing from the grid of dimension levels are allowed.

`additive_share` 1 minus `interaction_var` / `between_var`, and `interaction_share` the rest: a mean-square split, like a PCV computed from REML variances. It is near zero when the dimensions explain none of the differences between the strata, and can then fall below zero.

`per_dim` the variance of each dimension's level effects, with its number of levels minus one as divisor.

`mor` for a logit link, the median odds ratio between two distinct strata of the data: the exponential of the median of $|\zeta_j - \zeta_{j'}|$ over the pairs. With many strata it is computed from an evenly spaced subset of at least 1000 draws.

Each is the posterior median, with the equal-tailed credible interval at `conf_level` in the matching `*_ci` element. Every stratum counts once, whatever its size. `n_strata` is J. An `lme4` crossed-dimensions fit reports its variance components only.

Realised strata and contexts

For a `brms` fit with a `context`, `summary()$context$realised` and the "Realised strata and contexts" block of `print()` give the realised partition: over the posterior draws, the finite-population variances (Gelman 2005) of the effects of the strata and of each context's levels in the data. A context's variance component describes the distribution its levels were drawn from, and with few levels it follows the prior on its standard deviation; the variance of the effects of the levels in the data describes those levels, which the data pin down when each level has enough observations. With two contexts it is half the squared difference between their effects. Per draw, for a grouping factor with C levels in the data and effects u_k:

`var_stratum` the realised variance of the strata: of the stratum intercepts, or of the ζ_j of a crossed-dimensions fit (its `between_var`).

`per_context` each context's $\sum_k (u_k - \bar{u})^2/(C-1)$, and `context_var_total` their sum.

`other_var` the same, summed over any other random effect in the formula; given only when there is one.

`vpc_stratum` `var_stratum` over the sum of the realised variances and the within-stratum variance (see `vpc` in Realised strata), and `vpc_context` and `vpc_context_total` each context's and all contexts' share of that sum. Given for a Gaussian identity fit and for a binomial, Bernoulli or cumulative fit on its latent scale, not for a Poisson or negative-binomial fit. For a crossed-dimensions fit `vpc_stratum` is `decomposition$realised$vpc`.

Each is the posterior median, with the equal-tailed credible interval at `conf_level` in the matching `*_ci` element; for `per_context` and `vpc_context` a matrix with a row per context. Every level counts once, whatever its size. `n_strata` is the number of strata, and the partition's `context_levels` the number of each context's levels. `print()` of a two-model analysis shows the realised VPC and context share of the null model. These values depend little on the prior on the standard deviations when each context and stratum has enough data. With few observations per context each draw also carries the uncertainty about every context's effect, and they then move with the prior. An `lme4` fit reports its variance components only.

References

Gelman, A. (2005). Analysis of variance – why it is more important than ever. *The Annals of Statistics*, 33(1), 1-53. doi:10.1214/009053604000001048

See Also

`fit_maihda` for the single-model fitter, `compare_maihda_groups` for the group comparison, and `summary.maihda_model` for the variance summary.

Examples

```
data(maihda_health_data)
```

```
# One call: null + adjusted fit, VPC summary, and PCV decomposition. Writing the
# dimensions' additive main effects (Gender + Race) gives the fully-specified
# adjusted model; maihda() derives the null by dropping them.
a <- maihda(BMI ~ Age + Gender + Race + (1 | Gender:Race), data = maihda_health_data)
a                               # VPC (null) and PCV (null -> adjusted)
a$pcv                           # proportional change in between-stratum variance
a$formula                       # null:     BMI ~ Age + (1 | stratum)
a$adjusted_formula              # adjusted: null + Gender + Race main effects

# Omitting them is equivalent -- maihda() adds them to the adjusted model and
# emits a message; the null and PCV are identical to the explicit form above.
a0 <- maihda(BMI ~ Age + (1 | Gender:Race), data = maihda_health_data)

plot(a, type = "vpc")          # null model
plot(a, type = "effect_decomp")# adjusted model (additive vs intersectional)

# Crossed-dimensions decomposition: one model, the dimensions' main effects entered
# as RANDOM intercepts. The additive and interaction shares of the between-strata
# variance are read directly from the single fit (no null/adjusted pair).
cc <- maihda(BMI ~ Age + (1 | Gender:Race), data = maihda_health_data,
             decomposition = "crossed-dimensions")
cc                                  # VPC and additive/interaction shares
cc$decomposition$additive_share     # crossed-dimensions analogue of the PCV
cc$formula                          # BMI ~ Age + (1|Gender) + (1|Race) + (1|stratum)

# Add a higher-level grouping variable to also compare across its levels.
# maihda_country_data has a real country grouping (PISA achievement data):
data(maihda_country_data)
a2 <- maihda(math ~ 1 + (1 | gender:ses), data = maihda_country_data,
             group = "country")
a2
plot(a2, type = "group_vpc")
plot(a2, type = "group_pcv")

# Contextual cross-classified MAIHDA: instead of one model per country (group=),
# model the strata CROSSED with country in a single fit. The summary partitions
# the unexplained variance into between-stratum vs. between-country vs. residual,
# and the PCV is computed with country partialled out.
a3 <- maihda(math ~ 1 + (1 | gender:ses), data = maihda_country_data,
             context = "country")
a3
a3$summary$context$vpc_context_total  # the country (general contextual) share
plot(a3, type = "context_vpc")
```

---

maihda_auc *Area under the ROC curve (C-statistic), rank-based*

Description

Computes the AUC / C-statistic as the Mann-Whitney U statistic: the probability that a randomly chosen case (`y == 1`) is assigned a higher predicted value than a randomly chosen non-case (`y == 0`), with ties counting as one half. This needs no external package. An AUC of 0.5 is chance; 1 is perfect separation.

Usage

```
maihda_auc(prob, y)
```

Arguments

`prob`	Numeric vector of predicted probabilities (or any score where larger means more case-like).
`y`	Observed binary outcome as 0/1 numeric or logical, the same length as `prob`.

Value

A single number in `[0, 1]`, or `NA_real_` if either class is absent.

References

Merlo, J., Wagner, P., Ghith, N., & Leckie, G. (2016). An original stepwise multilevel logistic regression analysis of discriminatory accuracy: the case of neighbourhoods and health. *PLOS ONE*, 11(4), e0153778.

Examples

```
maihda_auc(c(0.1, 0.4, 0.35, 0.8), c(0, 0, 1, 1))
```

`maihda_country_data` *Cross-National Educational Achievement Data for MAIHDA*

Description

A cross-national dataset for demonstrating how Multilevel Analysis of Individual Heterogeneity and Discriminatory Accuracy (MAIHDA) can be used to compare intersectional inequality *across* a higher-level grouping variable (here, country) with `compare_maihda_groups` and `maihda`. Each row is a 15-year-old student; the intersectional strata are formed by `gender` and socioeconomic status (`ses`), and the outcome is the PISA mathematics score.

Usage

```
maihda_country_data
```

Format

A data frame with 3,600 rows (600 students in each of 6 countries) and 7 variables:

country Factor; one of Finland, Germany, United Kingdom, Italy, Japan, Mexico. The higher-level grouping variable.

gender Factor; student gender (female/male). A stratum dimension.

ses Factor; socioeconomic status as global tertiles (Low/Medium/High) of `escs`, computed on the pooled sample so a band means the same in every country. A stratum dimension.

escs Numeric; the PISA index of economic, social and cultural status (the continuous measure underlying `ses`).

math Numeric; PISA mathematics score (first plausible value). The primary outcome.

reading Numeric; PISA reading score (first plausible value).

low_math Factor; "Yes" if `math` is below 420 (PISA proficiency Level 2 baseline), else "No". A binary outcome for logistic examples.

Details

Intersectional inequality (the between-stratum share of variance, VPC/ICC) in mathematics achievement differs across the six countries, which is what makes the dataset a useful showcase for the group-comparison workflow.

The intersectional strata are `gender:ses` (2 x 3 = 6 strata). A canonical MAIHDA "null" model is `math ~ 1 + (1 | gender:ses)`; comparing its VPC across countries quantifies how much joint gender-by-class inequality in achievement varies between countries.

Note

This is a teaching/illustration dataset only. It uses a single PISA plausible value for each score and does **not** carry the PISA survey weights or complex sampling design, so results are **not** survey-representative and should not be used for substantive cross-national inference. (For your own survey data, the package supports design-weighted MAIHDA via the `sampling_weights` argument of `fit_maihda` / `maihda`.)

Source

Derived from the OECD Programme for International Student Assessment (PISA) 2018 student questionnaire data (OECD (2019), *PISA 2018 Database*), accessed and cleaned via the **learningtower** R package (MIT licensed), https://CRAN.R-project.org/package=learningtower. A balanced random subsample of 600 complete-case students per country was taken (seed 2026). The data preparation script is in `data-raw/maihda_country_data.R`.

Examples

```
data(maihda_country_data)

# Compare intersectional (gender x SES) inequality across countries
analysis <- maihda(
  math ~ 1 + (1 | gender:ses),
```

```
  data = maihda_country_data,
  group = "country"
)
analysis
plot(analysis, type = "group_vpc")
```

`maihda_cumulative` *Cumulative (ordinal) family marker for MAIHDA models*

Description

Specifies a cumulative (proportional-odds) model for an ordinal outcome in `fit_maihda` / `maihda`, with a choice of link: `maihda_cumulative("logit")` (the default, equivalent to `family = "ordinal"`) or `maihda_cumulative("probit")`. It plays the role a `stats` family object plays for the other families – there is no cumulative family constructor in `stats`, and using `brms::cumulative()` would require brms for a frequentist fit.

Usage

```
maihda_cumulative(link = c("logit", "probit"))
```

Arguments

`link`	The cumulative link: `"logit"` (default) or `"probit"`. These are the links for which the latent-scale VPC is defined (level-1 variance $\pi^2/3$ and 1 respectively).

Value

A family marker list with elements `family = "cumulative"` and `link`.

See Also

`fit_maihda`

Examples

```
maihda_cumulative()
maihda_cumulative("probit")
```

`maihda_describe` *Describe the MAIHDA sample before fitting*

Description

Builds the standard "Table 1" descriptives of a MAIHDA analytic sample *before* any model is fitted: the total and complete-case sample sizes, the observed vs. expected intersectional strata (and which are empty or small), the distribution of each stratum-defining dimension, family-aware per-stratum outcome summaries, missing-data accounting, and – for a contextual cross-classified design – the context units. It is the pre-model counterpart of `maihda_table`.

The description comes from the *same machinery* as the model: the strata are built by `make_strata` via the same formula-shorthand resolution as `fit_maihda`, the analytic sample reproduces the engines' own row selection (missing outcome, covariates, stratum dimensions, context, and – for a weighted design – missing/non-positive sampling weights), and the outcome family uses the same auto-detection as the fitters. Stratum IDs, labels, and counts therefore match a subsequent `make_strata()` / `fit_maihda()` / `maihda` on the same formula and data exactly.

Usage

```
maihda_describe(
  x,
  data = NULL,
  context = NULL,
  family = NULL,
  sampling_weights = NULL,
  flag_stratum_n = 20,
  include_empty_strata = TRUE,
  autobin = TRUE,
  digits = 3,
  weights = NULL
)
```

Arguments

`x`	A model formula (with the intersectional shorthand `outcome ~ covars + (1 | var1:var2)` or `... + (1 | stratum)` after `make_strata`), or a fitted `maihda_model` / `maihda_analysis` to describe post hoc.
`data`	A data frame with the model variables; required when `x` is a formula, and must be omitted for a fitted-model input.
`context`	Optional character vector naming higher-level context column(s), exactly as in `fit_maihda`; adds the per-unit `$context` table and the contextual identification checks. Must be omitted for a fitted-model input (taken from the fit).
`family`	`NULL` (default) auto-detects the family the same way the fitters do – a binary outcome becomes `"binomial"`, an ordered factor with 3+ levels becomes the cumulative (ordinal) model, anything else is described as `"gaussian"`. Otherwise

any family specification fit_maihda accepts (e.g. "poisson", "negbinomial", a family object). Must be omitted for a fitted-model input.

sampling_weights
: Optional single character string naming a numeric column of data with individual sampling (design) weights, as in fit_maihda; adds weighted counts and outcome summaries. Must be omitted for a fitted-model input.

flag_stratum_n
: Strata with n at or below this threshold are *flagged* as small (column small of $strata and a $warnings entry) – never dropped. This deliberately differs from make_strata()'s min_n, which drops. Default 20.

include_empty_strata
: Logical; when TRUE (default) the zero-count combinations of the observed dimension levels are appended to $strata with stratum = NA and n = 0.

autobin
: Passed to make_strata when the formula shorthand builds the strata, so the description matches a fit with the same autobin setting. Ignored for a fitted-model input (the strata are already built). Default TRUE.

digits
: Decimal places used by the print() method. Default 3.

weights
: Optional precision weights, given as a bare column name of data exactly as in fit_maihda, so the same call describes and fits the same sample. Rows whose weight is missing, zero, negative or non-finite are excluded from the analytic sample, as the engines exclude them. For a **binomial** model the weights also supply the denominator of R's second aggregated-binomial idiom (see ?glm), in which the trial counts ride in weights rather than in a cbind(successes, failures) response; the outcome is then summarised as events out of trials instead of one trial per row. That reading applies when the weights are non-unit whole numbers – the same rule maihda_discriminatory_accuracy uses, so the two never report different sample sizes for one model – and it covers a proportion response and a 0/1 frequency-cell response alike. Non-integral weights are not trial counts and are left alone. The binomial_weights overrides available on the AUC have no counterpart here: they select an estimand for the concordance, not a way to count a sample. Mutually exclusive with sampling_weights, which are design weights and mean something different. Must be omitted for a fitted-model input (the fit already carries its weights). Supply it by name.

Details

Family-aware outcome summaries. The per-stratum (and overall) outcome summary depends on the resolved family, and the applied summary type is recorded in $outcome_summary – a 0/1 outcome is never described with a Gaussian mean/SD as if continuous:

- gaussian (and other continuous families): mean, SD, median, min, max;
- binomial (including an aggregated outcome written either as cbind(success, failure) or, for brms, as success | trials(n)): event count, trials, and the observed proportion. Trials are the binomial denominator, so a row with a missing or non-positive trial count has no observed outcome;
- poisson / negative binomial: the same numeric summaries, read as the observed count mean (rate);

- cumulative (ordinal): the mean and (lower) median category score (categories scored 1..K in level order, the same convention as the package's plots), with the full category distribution in `$outcome_levels`.

Intersectional sparsity. `$overview` contrasts `n_strata_observed` with `n_strata_expected` (the full Cartesian product of the observed dimension levels) and `n_empty_strata`; with `include_empty_strata = TRUE` the empty combinations are enumerated as rows of `$strata` (with `stratum = NA`, so real stratum IDs stay identical to a subsequent fit). With many small strata the stratum effects are shrunk toward the mean and the variance components are imprecise with either engine.

Missing data. `$overview` separates the rows lost to a missing stratum dimension (`make_strata()` routes them to the NA stratum) from rows with a missing outcome, and reports the resulting complete-case `n_analytic` – computed with the same row mask the engines apply, so it equals `nrow(fit_maihda(...)$data)`.

Weights. With `sampling_weights`, weighted counts and weighted outcome means/proportions are reported alongside the unweighted ones, and rows with a missing or non-positive weight are excluded from the analytic sample (as the weighted engines do). As elsewhere in the package, these are population-weighted *point* summaries – not a complex survey design; no design-based variances are computed.

Units. `maihda_describe()` treats rows as the unit of description. For longitudinal (long-format) data a row is a measurement occasion, not a person; describing a fitted longitudinal model (`fit_maihda(id =, time =)`) additionally reports `n_individuals` and the median occasions per individual in `$overview`.

Fitted-model input. A `maihda_model` (or the `maihda_analysis` bundle from `maihda`) can be described post hoc; the formula, data, family, context, and sampling weights are taken from the fit, so the description covers the exact analytic sample the model used. For the `wemix`/`ordinal` engines the stored data already *is* the pre-filtered analytic sample, so total and analytic counts coincide there.

Value

An object of class `maihda_describe`: a list of export-ready data frames (pass to `write.csv()` or `knitr::kable()`) plus metadata:

`overview`	one row: `n_total`, `n_missing_outcome`, `pct_missing_outcome`, `n_rows_missing_dimensions` (rows outside every stratum), `n_analytic`, `n_strata_observed`, `n_strata_expected`, `n_empty_strata`; plus `n_invalid_weights`/`sum_weights` for a weighted description and `n_individuals`/`median_occasions_per_individual` for a fitted longitudinal model
`dimensions`	counts and percentages for each level of each MAIHDA dimension (`NULL` when the dimensions are unknown, e.g. a hand-built `stratum` column)
`strata`	one row per stratum: `stratum`, `label`, the dimension values, `n`, `n_analytic`, `n_missing_outcome`, `pct_missing_outcome`, the family-aware outcome summary columns, and the `small`/`empty` flags
`outcome_overall`	the same family-aware outcome summary over the whole sample
`outcome_levels`	the outcome's category distribution (binomial / ordinal outcomes; `NULL` otherwise)
`context`	one row per context unit: `context`, `level`, `n`, `n_analytic`, `pct` (`NULL` without a context)

`missingness` per variable (outcome, each dimension, each context, the weight column): `n_missing` and `pct_missing`

`warnings` data-quality flags as a data frame of `check` / `message` rows (zero rows when clean): auto-binned or ID-like or linear-numeric dimensions, empty/small strata, rows lost to missing dimensions, high or concentrated outcome missingness, and weakly identified contexts

`observations` a slim per-row frame (stratum and the outcome numerator/denominator on the summary scale) kept so `plot()` is self-contained

`outcome`, `family`, `family_detected`, `outcome_summary`, `event_level`, `strata_vars`, `context_vars`, `sampling_weights`, `source`, `engine`, `longitudinal`, `response_recoding`, `call`, `...`
metadata used by `print()`

See Also

`maihda_table` for the post-model tables, `make_strata`, `fit_maihda`, `plot` for the descriptive plots.

Examples

```
data(maihda_health_data)
desc <- maihda_describe(BMI ~ Age + (1 | Gender:Race:Education),
                        data = maihda_health_data, flag_stratum_n = 20)
desc
desc$overview
head(desc$strata)

# Binary outcome: auto-detected binomial, proportions instead of means
desc_bin <- maihda_describe(Obese ~ (1 | Gender:Race:Education),
                            data = maihda_health_data)
desc_bin$outcome_levels

# Contextual cross-classified design (country as context)
data(maihda_country_data)
desc_ctx <- maihda_describe(math ~ escs + (1 | gender:ses),
                            data = maihda_country_data, context = "country")
desc_ctx$context

# Export-ready tables
# write.csv(desc$strata, "table1_strata.csv", row.names = FALSE)
```

`maihda_discriminatory_accuracy`

Discriminatory accuracy of a binary MAIHDA model

Description

Bundles the individual-level discriminatory-accuracy summaries for a binomial MAIHDA model: the AUC / C-statistic (how well the model's predicted probabilities separate cases from non-cases) and the Median Odds Ratio. Applied to a strata-only (null) model, the AUC is the discriminatory accuracy of the intersectional strata themselves – Merlo's central quantity; comparing it with an adjusted model shows whether individual covariates beyond stratum membership sharpen classification. The AUC is computed for any binomial link; the Median Odds Ratio is reported only for the logit link and is `NA` otherwise (e.g. for a probit fit), since the MOR is an odds-ratio-scale quantity.

The AUC is apparent (in-sample). It scores the same observations used to estimate the model – the fixed effects, the variance components, and the shrunken stratum BLUPs – so it is an *apparent* (resubstitution) AUC and is optimistically biased, the more so with small or sparse strata. It is reported as the conventional descriptive MAIHDA discriminatory accuracy (Merlo 2018), **not** as a cross-validated estimate of out-of-sample predictive discrimination; the returned object carries `apparent = TRUE` and `print()` says so. For genuine predictive accuracy, validate with an out-of-fold (group-aware) scheme or an optimism correction.

Aggregated-binomial fits are supported in all three forms R accepts – an lme4 `cbind(success, failure)` response, an lme4 response in [0, 1] whose trial counts are supplied as `weights =` (see `glm`), and a brms `y | trials(n)` response: the AUC is the count-weighted C-statistic over the implied individual-level 0/1 data, and `n_case` / `n_control` are the total successes / failures. All three spellings of the same data give the same AUC. This includes individual 0/1 records collapsed to frequency cells (every row all-success or all-failure), whose weights are trial counts like any others; see `binomial_weights` to override that reading.

Scope. When the model carries random effects *beyond* the intersectional partition – a contextual `(1 | school)` from `fit_maihda(context = )` or an explicit extra grouping such as `(1 | site)` – the headline `auc` is the *intersectional-scope* concordance: it **excludes** those other random effects but keeps the fixed effects plus the stratum random effect (and, for a crossed-dimensions fit, the additive dimension effects). The concordance of the full model including the other random effects is reported separately as `auc_full` (`auc_scope = "intersectional"`). **Caveat – this is not strata-only discrimination.** The intersectional-scope score retains the *entire* fixed-effects predictor, so when the model is adjusted for individual-level covariates (e.g. `age`, `income` that vary *within* strata) those covariates enter this AUC too; it is the concordance of the adjusted fixed effects plus the intersectional random effect(s), not of the strata alone, and it then matches the between-stratum MOR's scope only when the fixed part is intercept-only. For a strata-only discriminatory accuracy, score the null (strata-only) model. For the canonical single-`(1 | stratum)` model with no other random effects, the full and intersectional scopes coincide and only `auc` is reported (`auc_scope = "model"`, `auc_full` absent).

Usage

```
maihda_discriminatory_accuracy(
  model,
  binomial_weights = c("auto", "trials", "analytic")
)
```

Arguments

`model`	A `maihda_model` from `fit_maihda` fitted with a `binomial` family – including an aggregated response (an lme4 `cbind(success, failure)` or a brms `y |`

trials(n)) – or the bernoulli family a binary 0/1 outcome is fit with under engine = "brms".

binomial_weights

How to read non-unit weights = on an **lme4** binomial fit when computing the AUC. "auto" (default) reads integral weights as **trial counts**, which is what R documents a binomial prior weight to be (glm: "For a binomial GLM prior weights are used to give the number of trials"), and leaves non-integral weights – which cannot be counts – on the observation-level path, flagged precision_weights_ignored. "trials" forces the trial-count reading even for non-integral weights (the fractional case/control mass is carried as it stands, with a warning). "analytic" forces the ordinary observation-level concordance, ignoring the weights and setting precision_weights_ignored = TRUE; it is an error for a response carrying values strictly between 0 and 1, which has no observation-level case/control reading. Has no effect on a cbind(success, failure) response (its denominator is structural), on a brms y | trials(n) fit, or on a design-weighted (sampling_weights) fit.

Value

An object of class maihda_da: a list with auc, auc_scope, auc_full, mor, n_case, n_control, family, link, engine, weighted, weight_type, precision_weights_ignored and apparent (always TRUE – the AUC is in-sample; see Description). mor is NA for a non-logit binomial link, where the AUC is still reported. For an aggregated-binomial fit n_case / n_control are the total successes / failures. weighted is TRUE only when the AUC is a genuinely weighted (population-mass) concordance, with weight_type "sampling" for a design-weighted fit (each observation contributes its sampling weight as case/control mass, estimating the population discriminatory accuracy); n_case / n_control stay unweighted observation counts. weight_type is NULL for an unweighted AUC. An **aggregated-binomial** fit is reported weighted = FALSE with weight_type = NULL: its count-weighted AUC equals the ordinary individual-level concordance over the implied 0/1 data (the trial counts are real observations, not sampling weights), so it is not a design-weighted population quantity. Aggregation is recognised from an lme4 cbind(success, failure) matrix response, from non-unit integral lme4 weights= on a response in [0, 1] (the trial counts, see glm), or from a brms y | trials(n) term. The weights-based form covers a response of exactly 0/1 – individual records collapsed to frequency cells – because for the binomial family a prior weight *is* a trial count: the weighted log-likelihood of one 0/1 row is exactly that of w trials sharing its outcome, so the fit equals the row-expanded one and its AUC should too. Pass binomial_weights to override. When a response times its trial counts is *not* a whole number of successes – a malformed binomial, which glmer itself warns about as "non-integer #successes" – the fractional case/control mass is carried into the AUC as it stands, with a warning, rather than rounded into whole observations that were never collected; n_case / n_control are then fractional. **Weights not read as trial counts** – non-integral weights=, or any weights under binomial_weights = "analytic" – are ignored by the AUC, which is then the ordinary observation-level concordance (weighted = FALSE), with precision_weights_ignored = TRUE flagging that such weights were present.

References

Merlo, J. (2018). Multilevel analysis of individual heterogeneity and discriminatory accuracy (MAIHDA) within an intersectional framework. *Social Science & Medicine*, 203, 74-80.

See Also

maihda_auc, maihda_mor

Examples

```
# Obese (Yes/No) by intersectional strata of Gender x Race
strata <- make_strata(maihda_health_data, vars = c("Gender", "Race"))
d <- maihda_health_data
d$stratum <- strata$data$stratum
m <- fit_maihda(Obese ~ (1 | stratum), data = d, family = "binomial")
maihda_discriminatory_accuracy(m)
```

---

maihda_glance *Glance at a MAIHDA model or analysis*

Description

glance methods that return the MAIHDA headline as a one-row tibble: the variance partition coefficient (VPC/ICC), and – for a maihda_analysis – the proportional change in variance (PCV), plus the additive/interaction shares and the discriminatory accuracy (AUC, MOR) for a binomial outcome. The layout is uniform across the lme4, brms, WeMix and ordinal engines. No other package emits this row from the underlying fit, because PCV needs the null+adjusted pair that only a maihda_analysis holds.

Usage

```
## S3 method for class 'maihda_summary'
glance(x, ...)

## S3 method for class 'maihda_model'
glance(x, ...)

## S3 method for class 'maihda_analysis'
glance(x, ...)
```

Arguments

x	A maihda_summary, maihda_model, or maihda_analysis.
...	Unused, for S3 consistency.

Value

A one-row tibble. glance.maihda_analysis adds pcv (with pcv.conf.low/pcv.conf.high when bootstrapped; a brms PCV is a point estimate), the adjusted-model auc.adjusted, nobs, family and mode to the columns produced for a single summary.

See Also

maihda_tidiers for the per-estimate tidy() methods.

Examples

```
data("maihda_health_data")
a <- maihda(BMI ~ Age + Gender + Race + Education + (1 | Gender:Race:Education),
            data = maihda_health_data)
glance(a)
```

---

maihda_health_data *NHANES Health Data Subset for MAIHDA Use*

Description

A pedagogical subset of the National Health and Nutrition Examination Survey (NHANES) dataset, serving as a real-world example for Multilevel Analysis of Individual Heterogeneity and Discriminatory Accuracy (MAIHDA). Contains selected records demonstrating intersectional demographic health inequalities.

Usage

```
maihda_health_data
```

Format

A data frame with 3,000 rows and 7 variables:

BMI Body Mass Index (kg/m^2), a continuous outcome variable.

Obese Factor indicating obesity status (No/Yes).

Age Age in years at screening, a continuous covariate.

Gender Gender of the participant (male/female).

Race Self-reported race/ethnicity.

Education Educational attainment level.

Poverty Poverty to income ratio, a continuous covariate. Some values may be missing.

Note

This is a teaching/illustration dataset only. It is a non-random subsample and does **not** carry the NHANES survey weights or complex sampling design, so results are **not** survey-representative and should not be used for substantive population inference. (For your own survey data, the package supports design-weighted MAIHDA via the sampling_weights argument of fit_maihda / maihda.)

Source

Derived from the NHANES R package. Original data collected by the Centers for Disease Control and Prevention (CDC).

Examples

```
data(maihda_health_data)

# Example usage:
# strata_result <- make_strata(maihda_health_data, vars = c("Gender", "Race", "Education"))
# model <- fit_maihda(BMI ~ Age + (1 | stratum), data = strata_result$data)
```

---

maihda_ic *Information criteria for MAIHDA models*

Description

Reports the relative-fit information criteria for one or more MAIHDA models, to help choose between model *structures* (different covariate sets, strata definitions, or families) – a question the VPC/ICC and PCV do not address. The criteria reported depend on the engine: **AIC** and **BIC** for the likelihood engines (lme4, and ordinal::clmm), and the Bayesian **WAIC** and **LOOIC** (leave-one-out information criterion) for brms. Lower is better for all four.

Usage

```
maihda_ic(..., model_names = NULL)
```

Arguments

...	One or more maihda_model objects (from fit_maihda) or maihda_analysis objects (from maihda). A maihda_analysis contributes its null model and, when present, its adjusted model as separate rows.
model_names	Optional character vector of names, one per ... argument. A maihda_analysis argument's null/adjusted rows are suffixed from its name.

Details

REML vs ML. lmer fits Gaussian models by REML by default, and a REML log-likelihood (hence its AIC/BIC) is *not* comparable across models with different fixed effects – exactly the canonical MAIHDA null-vs-adjusted comparison. When more than one model is supplied, maihda_ic() therefore refits any REML lmer model with maximum likelihood (refitML) before computing AIC/BIC, matching the behaviour of anova() on lme4 models; the estimator column records when this happened. For a single model the criterion is reported as fitted (the estimator column then reads "REML").

Comparability. Like the VPC, information criteria are only comparable across models fitted to the *same* analytic sample (same rows and outcome) with the *same* weights – prior (precision) weights

and sampling (design) weights each change which likelihood, or pseudo-likelihood, is being maximised, so the criteria of a weighted and an unweighted fit of the identical model are not on a common scale. Unlike the VPC and PCV, they do not always need the same family: a criterion compares log-likelihoods of the same observations, so Poisson and negative-binomial fits of the same counts are comparable, as are binomial or cumulative (ordinal) fits that differ in their link, and a delta is reported for them. Any other change of family or link withholds it – a Gaussian vs a Poisson fit compares a density with a probability, and the continuous families are compared only within the same family and link. Criteria are never comparable between the likelihood engines and `brms` (AIC/BIC vs WAIC/LOOIC are different scales). When the supplied models differ in any of these respects `maihda_ic()` warns and omits the `delta` column, still reporting each model's own criteria.

Parameter count. `df` counts the estimated parameters. A negative-binomial `lme4` fit with a fixed `theta` (`family = MASS::negative.binomial(theta)`) therefore has one fewer than `lme4`'s own count, which includes theta for every negative-binomial family, so its `AIC` is 2 and its `BIC` `log(n)` below `AIC()` and `BIC()` on the fitted model; `glm()` counts that family the same way. A `family = "negbinomial"` fit estimates theta and counts it.

lme4 GLMM likelihoods. The criteria of a quadrature fit (`nAGQ > 1`) are the complete likelihood and compare with a Laplace fit's: where `lme4`'s `logLik()` of such a fit leaves out the saturated log-likelihood, `maihda_ic()` adds it back. A `glmer` fit of a family with a scale parameter – a Gaussian with a non-identity link, Gamma, inverse Gaussian – reports `NA` criteria (estimator `"ML (glmer scale family: no likelihood)"`): `lme4`'s `logLik()` for those families is not the marginal likelihood, and on test fits it exceeded the largest value that likelihood attains.

Predictive target of the Bayesian criteria. `brms::waic()` and `brms::loo()` are computed from pointwise log-likelihoods *conditional on the fitted random effects*, so WAIC/LOOIC assess prediction of new observations *within the strata (and persons or contexts) already represented in the data* – not performance for a new, unseen intersectional stratum. Choosing between strata definitions on LOOIC therefore compares conditional predictive fit; generalisation to new strata is a leave-one-group-out cross-validation question (e.g. `brms::kfold()` with `group = "stratum"`), which this package does not wrap. AIC/BIC for the likelihood engines are instead computed from the *marginal* likelihood (random effects integrated out) – a further reason the likelihood and Bayesian criteria are never comparable with each other.

Design-weighted fits. For the `wemix` (design-weighted) engine the criteria are reported as `NA`: a pseudo-likelihood with sampling weights does not define a standard AIC/BIC. A `brms` fit with `sampling_weights` is treated the same way: its sampling weights enter as likelihood weights, giving a pseudo-posterior whose weighted pointwise log-likelihoods are not log predictive densities, so WAIC/LOOIC are likewise reported as `NA` (the `estimator` column reads `"Bayesian (weighted pseudo-posterior)"`).

Value

A `data.frame` of class `maihda_ic` with one row per model and the columns that apply: `model`, `n` (analytic sample size), `estimator`, `df` (number of parameters; likelihood engines), `logLik`, `AIC`, `BIC` (likelihood engines), `WAIC`, `LOOIC` (brms), and – when more than one model is supplied – `delta` (the difference from the best model on the primary criterion: AIC for the likelihood engines, LOOIC for brms). Columns that are entirely `NA` across the supplied models are dropped.

See Also

compare_maihda, which reports these criteria alongside the VPC/ICC, and calculate_pcv for the variance decomposition.

Examples

```
strata <- make_strata(maihda_sim_data, vars = c("gender", "race"))
null_model <- fit_maihda(health_outcome ~ 1 + (1 | stratum), data = strata$data)
adj_model  <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata$data)

# AIC/BIC for two nested structures (REML lmer fits are ML-refitted first)
maihda_ic(null_model, adj_model, model_names = c("Null", "Adjusted"))

# Or straight from a one-call maihda() analysis (null + adjusted rows)
a <- maihda(health_outcome ~ age + gender + race + (1 | gender:race),
            data = maihda_sim_data)
maihda_ic(a)
```

---

maihda_interactions *Flag strata with credibly non-zero intersectional interaction*

Description

Reports, for each intersectional stratum, the **interaction** component of its outcome – the stratum random effect (BLUP) of an *adjusted* MAIHDA model, i.e. how far the stratum departs from the additive main-effects prediction of its defining dimensions – and **flags** the strata whose interaction is credibly different from zero. This is the heart of "where is there genuine intersectionality": a flagged stratum is one whose outcome departs credibly from what the additive parts predict – a descriptive statement about the stratum's outcome, not a causal claim about identity.

Usage

```
maihda_interactions(
  object,
  conf_level = 0.95,
  adjust = "BH",
  rope = NULL,
  scale = c("link", "response"),
  ...
)
```

Arguments

object	A maihda_analysis from maihda (preferred – its adjusted / crossed-dimensions model is used automatically) or a maihda_model from fit_maihda (which should be the *adjusted* model; a null model is accepted but warned about).

`conf_level` Confidence / credible level for the interval and the flag. Default 0.95.

`adjust` Multiple-comparison adjustment for the per-stratum p-values (frequentist engines only): `"BH"` (default; false-discovery rate) or any method accepted by `p.adjust`, including `"none"` for the uncorrected, per-stratum individual-testing view. Ignored for `brms` (which uses the posterior tail directly; a message is shown only if you set it explicitly).

`rope` Optional equivalence region (a "smallest interaction of interest") for an "is the interaction *negligible*?" reading (Schuirmann 1987; Kruschke 2018), read on the requested `scale` – log-odds under the default, probability points under `scale = "response"` for a logistic fit. `NULL` (default) gives only the usual zero-centred flag. A single positive number `d` means the symmetric region `c(-d, d)`; or supply `c(lower, upper)`. When set, the result gains a `decision` column classifying each stratum from its `conf_level` interval relative to the region: `"relevant"` (interval entirely outside it), `"negligible"` (entirely inside it), or `"inconclusive"` (straddling a bound).

`scale` Scale the interaction is reported on. `"link"` (default) gives the stratum BLUP u_j on the model's link scale – for a Gaussian fit the outcome's own units, for a logistic fit a log-odds departure. `"response"` instead gives Evans et al.'s (2024, sec. 2.5.1) $\pi_j^B = \pi_j - \pi_j^A$: the stratum's total predicted outcome minus the outcome implied by its additive main effects alone, so a logistic fit reports a difference in *probability*, a count fit a difference in expected count, and a cumulative (ordinal) fit a difference in expected category score – the printed column is named for whichever it is. The two scales flag the same strata (see the section below); an identity-link fit returns the same numbers either way. For a `decomposition = "crossed-dimensions"` model the additive baseline is the dimension random effects, matching that decomposition.

`...` Currently unused.

Details

It must be read off the adjusted model. Only when the dimensions' additive main effects are in the model (the *adjusted* model of the two-model decomposition, or the crossed-dimensions model) does the stratum random effect isolate the *pure interaction*. On a null model the stratum random effect is the total between-stratum deviation (additive + interaction), so passing one is flagged with a warning. The opposite mis-specification is flagged too: a model that adds a *fixed* interaction among the dimensions (e.g. `var1 * var2`) absorbs the intersectional effect into fixed cell means, so the stratum random effect is no longer the pure interaction. Passing a `maihda` result uses the right model automatically.

Frequentist vs. Bayesian evidence. For the frequentist engines (`lme4`, `wemix`, `ordinal`) the flag comes from the BLUP's conditional standard error: a Wald interval at `conf_level` and a two-sided p-value, with an optional multiplicity correction (`adjust`). For `brms` the full posterior is already available, so the *exact* posterior tail is used – a credible interval at `conf_level` and the probability of direction `pd = max(P(BLUP > 0), P(BLUP < 0))` (in `[0.5, 1]`; the sign is in `direction`) – and `adjust` is not applied: the posterior is already partially pooled, the hierarchical-shrinkage answer to multiplicity (Gelman, Hill & Yajima 2012). That is a per-stratum answer; the marginal intervals carry no formal joint error-rate guarantee for the collective claim "an interaction exists *somewhere*" (the disjunction reading below applies to Bayesian flags too).

Multiplicity: partial pooling and a correction are different things, and the experts disagree.

- *Shrinkage (magnitude/sign).* The stratum BLUP is partially pooled, so extreme values are regularised toward the grand mean, attenuating exaggerated-magnitude and wrong-sign (Type M/S) error (Gelman & Carlin 2014). Gelman, Hill & Yajima (2012) argue this shrinkage *usually substitutes* for a classical multiple-comparisons correction (the problem can "disappear entirely" in the hierarchical model); on that view the flag/no-flag step itself is what to avoid – the null of an *exactly* zero interaction is rarely the question (McShane, Gelman et al. 2019) – so report the estimate and its interval.
- *Whether to correct.* If you do want an error-rate screen, whether a correction is warranted depends on the *inferential structure* of the claim. Each stratum as its own pre-specified hypothesis ("does *this* stratum interact?") is *individual* testing and needs none – **only** if you do not also read the flags collectively. Once the question is "is there an interaction *somewhere*?" – which an automated all-strata scan effectively is – it is *disjunction* testing and a correction applies.

`adjust = "BH"` **is the default**: fitting and flagging every stratum in one call is the disjunction/screening case, where controlling the expected *proportion* of false discoveries (FDR) is the appropriate goal. Pass `adjust = "none"` only when each stratum is a genuine, pre-specified individual hypothesis. The FDR choice (over family-wise `"bonferroni"`/`"holm"`) is this package's, matching that screening goal. The flag itself is a Wald test on a shrunken BLUP whose conditional SE treats the variance components as known, so it (and any `adjust` on it) is an explicit, *conservative* screen – strict, not liberal. Partial pooling deflates a truly-null stratum's BLUP more than its conditional SE, so the null z-statistic is sub-normal (variance about the shrinkage fraction, below 1) and the screen under-flags rather than over-flags, degenerating to no flags at the singular/zero-variance boundary. Lead with the interval (and, for `brms`, the probability of direction); the substantive question is often not whether an interaction differs from zero but whether it exceeds a smallest interaction of interest (an equivalence reading; Schuirmann 1987; Kruschke 2018), read from the interval.

The interaction is reported on the model's link (latent) scale – a log-odds deviation for a logistic model, etc. – because the additive/interaction split is only exact there. On a non-identity link that makes it a *multiplicative* departure, which is a different quantity from a shift in the outcome itself; see the section below before reporting it.

Value

An object of class `maihda_interactions` (a data frame), one row per stratum, sorted flagged-first then by `abs(interaction)`. Columns common to every engine: `stratum`, `label`, `n` (stratum size), `interaction` (the BLUP, under the default `scale`), `lower`/`upper` (the interval), `flagged` (logical), and `direction` (`"above"`/`"below"` the additive expectation). Under `scale = "response"` the `interaction`/`lower`/`upper` columns hold π_j^B and its interval, and `se` is dropped: the conditional standard error is a link-scale quantity, and the response-scale interval is deliberately not symmetric about the estimate, so no single SE would reproduce it. Frequentist fits add `se` (link scale only) and `p_value` (and `p_adjusted` when `adjust != "none"`). `p_value` is a *conditional* screening statistic – a Wald tail on the shrunken BLUP's conditional SE, with the variance components treated as known – **not** a calibrated frequentist p-value; it is *conservative* (stochastically large under a true null), so the BH-adjusted flag under-flags truly-null strata rather than delivering an exact error-rate guarantee (see Details). `brms` instead adds `pd` (probability of direction, `max(P(>0), P(<0))` in `[0.5, 1]`). When `rope` is set, a `decision` column (`"relevant"`/`"negligible"`/`"inconclusive"`) is added. Attributes record `conf_level`, `adjust`, `rope`, `engine`, `model_type`, `n_strata`, `n_flagged`, `scale`,

`link` (the model's link name, so a caller can tell a log-odds departure from one in the outcome's own units), `response_kind` (what the model's response scale is – `"probability"`, `"count"`, `"score"` or `"response"` – which the link alone does not settle, since a cumulative fit is logit-linked but scores categories), `singular` (a variance of the fit is at zero, in any term), `interaction_at_boundary` (the variance of the term the interaction is read from is at zero, so its standard errors collapse; `print()` then says so), and – on a non-identity link under the default `scale` – `response_interaction`, the outcome-scale estimates keyed by stratum. That last attribute exists so `print()` can show the outcome-scale size beside the link-scale one; the columns are the same either way, and `scale = "response"` is how to obtain those numbers, with their interval, as data.

What the interaction means on a non-identity link

For a Gaussian (identity-link) fit the stratum BLUP u_j *is* the part of the stratum's mean outcome attributable to interaction, in the outcome's own units. For a logistic fit it is not: u_j is a deviation in **log-odds**, so what this function reports is the *multiplicative*, odds-scale interaction. (What follows is written for the logistic case, the one the literature works in. The same logic holds on any non-identity link, in its own units: a Poisson fit's u_j is a log-rate departure and `scale = "response"` gives it as a difference in expected counts, and a cumulative (ordinal) fit's is a latent shift reported as a difference in expected category score. The printed column is named for whichever applies.) Evans et al. (2024, section 2.5.1) put it directly – in a logistic MAIHDA one can "no longer directly interpret u_j as the change in mean outcomes (i.e., shift in probabilities) attributable to interaction effects".

Three quantities, of which this function reports two. Keeping them apart is the whole of the difficulty:

1. **The multiplicative interaction,** u_j – the departure from additivity *in log-odds*, i.e. whether a dimension multiplies the odds by the same factor at every level of the others. This is what the default `scale = "link"` reports and what `flagged` is about.
2. **The same departure in outcome units,** π_j^B – what `scale = "response"` returns. Evans et al. (2024) define it as

$$\pi_j^B = \pi_j - \pi_j^A, \qquad \pi_j = \text{logit}^{-1}(x_j'\beta + u_j), \qquad \pi_j^A = \text{logit}^{-1}(x_j'\beta),$$

   the gap between a stratum's total predicted probability and the probability implied by the additive main effects alone, and rank-plot π_j^B where the linear case plots u_j. This is quantity 1 *re-expressed*, not a second finding: it is zero exactly when u_j is zero and it flags the same strata, so it changes the units you report, not what you may conclude.
3. **The additive (risk-difference) interaction** – whether a dimension adds the same number of percentage points of risk at every level of the others. **Neither of the above reports this**, and it is generally non-zero even where u_j is exactly zero, because the logistic curve is steeper in the middle than in the tails. With a -2 baseline and $+0.7$ for each of two dimensions and no interaction at all ($u_j = 0$ throughout), the second dimension still adds 9.5 percentage points of risk at one level of the first and 14.0 at the other. That excess is a property of the link, not a finding.

So "no strata flagged" supports "no credible **multiplicative** interaction", not "no interaction": additivity in log-odds does not carry over to probabilities, so quantity 3 is generally non-zero regardless, and it is often the one a policy audience cares about. (*Generally*, not always. The risk-difference interaction is positive below the curve's midpoint and negative above it, so it passes through zero:

with two dimensions of effect a and b it vanishes exactly at an intercept of $-(a+b)/2$, where the two comparisons straddle the midpoint symmetrically. That is a single configuration, not the general case, and the flags say nothing about which one you are in either way.)

Both scales flag the same strata. Writing $g(u)$ for the map from a stratum's BLUP to its π_j^B, g is strictly increasing with $g(0) = 0$, so estimate, interval endpoints and zero all carry across together. `flagged`, `direction`, `p_value` / `p_adjusted` and `pd` are therefore identical under either `scale`, and the response-scale interval is the *exact* image of the link-scale one rather than a delta-method approximation – inheriting its conditionality (fixed effects and variance components held at their point estimates), which is why no simulation step is needed. What does change is the size and the ranking: the same log-odds departure is worth more probability near $\pi = 0.5$ than in the tail, so strata are ordered by the quantity actually reported.

If quantity 3 is what you need, no argument here will give it to you – fit a linear probability model instead: `fit_maihda(..., family = "gaussian")` on a 0/1 outcome, which `fit_maihda` signposts when it auto-detects a binary outcome. There the BLUP *is* the risk-difference interaction by construction, at the usual costs (predictions outside `[0, 1]`, heteroskedastic residuals).

References

Evans, C. R., Williams, D. R., Onnela, J. P., & Subramanian, S. V. (2018). A multilevel approach to modeling health inequalities at the intersection of multiple social identities. *Social Science & Medicine*, 203, 64-73.

Merlo, J. (2018). Multilevel analysis of individual heterogeneity and discriminatory accuracy (MAIHDA) within an intersectional framework. *Social Science & Medicine*, 203, 74-80.

Evans, C. R., Leckie, G., Subramanian, S. V., Bell, A., & Merlo, J. (2024). A tutorial for conducting intersectional multilevel analysis of individual heterogeneity and discriminatory accuracy (MAIHDA). *SSM - Population Health*, 26, 101664. doi:10.1016/j.ssmph.2024.101664

Gelman, A., Hill, J., & Yajima, M. (2012). Why we (usually) don't have to worry about multiple comparisons. *Journal of Research on Educational Effectiveness*, 5(2), 189-211.

Gelman, A., & Carlin, J. (2014). Beyond power calculations: assessing Type S (sign) and Type M (magnitude) errors. *Perspectives on Psychological Science*, 9(6), 641-651.

Rubin, M. (2021). When to adjust alpha during multiple testing: a consideration of disjunction, conjunction, and individual testing. *Synthese*, 199(3-4), 10969-11000. doi:10.1007/s11229021-032764

McShane, B. B., Gal, D., Gelman, A., Robert, C., & Tackett, J. L. (2019). Abandon statistical significance. *The American Statistician*, 73(sup1), 235-245.

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics*, 15(6), 657-680.

Kruschke, J. K. (2018). Rejecting or accepting parameter values in Bayesian estimation. *Advances in Methods and Practices in Psychological Science*, 1(2), 270-280.

See Also

`maihda`, `calculate_pcv`, `summary.maihda_model`; and `plot(..., highlight_interactions = TRUE)` to mark the flagged strata on the effect-decomposition / predicted / shrinkage plots.

Examples

```
data(maihda_health_data)
a <- maihda(BMI ~ Age + Gender + Race + (1 | Gender:Race),
            data = maihda_health_data)
maihda_interactions(a)                  # FDR-screened (default adjust = "BH")
maihda_interactions(a, adjust = "none") # uncorrected per-stratum individual view
maihda_interactions(a, rope = 0.1)      # equivalence: |interaction| within 0.1?

# On a logistic fit the BLUP is a log-odds departure; scale = "response" reports
# Evans et al.'s (2024) pi_j - pi^A_j, the same interaction in probability points.
b <- maihda(Obese ~ Gender + Race + (1 | Gender:Race),
            data = maihda_health_data, family = "binomial")
maihda_interactions(b, scale = "response")
```

---

maihda_long_data *Simulated Longitudinal Data for MAIHDA*

Description

A simulated long-format (one row per person-occasion) panel for demonstrating **longitudinal / growth-curve MAIHDA** (`fit_maihda(id =, time =)` and `maihda(decomposition = "longitudinal")`). 600 individuals are each measured over five waves on a continuous wellbeing score, within 12 intersectional strata defined by gender x ethnicity x education. The between-stratum *trajectory* differences are constructed to be mostly additive (each dimension's main effect on the baseline level and on the rate of change) with one genuine multiplicative interaction, so the longitudinal PCV – a high but sub-1 `PCV_slope` – is demonstrable.

Usage

```
maihda_long_data
```

Format

A data frame with 3000 rows (600 persons x 5 waves) and 8 variables:

id Person identifier (level 2); repeated across waves.

wave Measurement occasion, 0 to 4 (the numeric time variable).

gender Gender (`Women`/`Men`); a stratum dimension.

ethnicity Ethnicity (`EthA`/`EthB`/`EthC`); a stratum dimension.

education Education (`Low`/`High`); a stratum dimension.

age Baseline age in years, a time-invariant covariate.

wellbeing Continuous wellbeing outcome (the growth-curve response).

low_wellbeing Binary companion outcome (1 = bottom 40% of wellbeing), for the logistic longitudinal path.

Source

Simulated for the purpose of the MAIHDA package. The growth structure follows the longitudinal MAIHDA of Bell, Evans, Holman & Leckie (2024) doi:10.1016/j.socscimed.2024.116955.

Examples

```
data(maihda_long_data)

# Time-varying VPC from a 3-level growth model:
m <- fit_maihda(wellbeing ~ wave + (1 | gender:ethnicity:education),
                data = maihda_long_data, id = "id", time = "wave")
summary(m)

# Additive-vs-multiplicative PCV (null vs adjusted growth model):
a <- maihda(wellbeing ~ wave + (1 | gender:ethnicity:education),
            data = maihda_long_data, id = "id", time = "wave",
            decomposition = "longitudinal")
a$pcv
```

`maihda_mor` *Median Odds Ratio (MOR) for a logistic MAIHDA model*

Description

The Median Odds Ratio translates the between-stratum variance of a logistic MAIHDA model onto the odds-ratio scale: the median relative change in the odds of the outcome when comparing two individuals from randomly chosen strata (higher- vs lower-risk). `MOR = exp(sqrt(2 * V_A) * qnorm(0.75))`, where `V_A` is the between-stratum (latent, logit-scale) variance. An MOR of 1 indicates no between-stratum heterogeneity. The MOR is defined only for the **logit** link (it is the median *odds* ratio); a non-logit binomial fit such as `probit` is rejected, because its latent variance is on a different scale and the `exp(...)` above would not be an odds ratio.

For a **cumulative-logit** (ordinal) MAIHDA model the same formula applies to the latent logit-scale between-stratum variance and is the *median cumulative odds ratio*: the median relative change in the odds of being at or below any given outcome category between two randomly chosen strata (under the model's proportional-odds assumption it is the same for every category split).

Scope. `V_A` is the *between-stratum* variance. Contextual (`context = `) and other non-stratum random effects are never included: the MOR compares two individuals from randomly chosen strata *within the same context*.

Crossed-dimensions fits use a different calculation. The closed form above assumes the two strata's random effects are *independent*, which is what makes their difference $N(0, 2V_A)$. In a crossed-dimensions fit (from `maihda(decomposition = "crossed-dimensions")`) a stratum's effect is the sum of its dimension effects plus the intersection effect, so two strata sharing a dimension – say two strata that are both "female" – share that dimension's random effect and are **correlated**. Their difference is then a *mixture* of normals, one component per pattern of shared dimensions, and applying the closed form to the summed variance overstates the MOR (substantially so when the variance sits mainly in the additive dimensions, and not at all when it sits entirely in the interaction).

The MOR reported for such a fit is therefore computed from that mixture, under an explicit sampling scheme: **two distinct strata drawn uniformly at random from the strata present in the analytic sample**. Writing τ_d^2 for dimension d's variance and τ_I^2 for the intersection variance, a pair differing on the dimension set D^* has difference variance $v = 2(\tau_I^2 + \sum_{d \in D^*} \tau_d^2)$, and the MOR is `exp(x)` for the `x` solving $\sum_{pairs}(2\Phi(x/\sqrt{v}) - 1)/n_{pairs} = 0.5$. For a canonical single-stratum fit the two calculations coincide, and that closed form is used.

brms fits. For a `brms` fit `V_A` is the posterior median of the between-stratum variance, the value in the variance-components table of `summary.maihda_model`. The closed form is monotone in `V_A`, so the MOR is then the posterior median of the Median Odds Ratio itself. A crossed-dimensions fit evaluates the mixture at the posterior median of each variance. With few strata the posterior of a variance is right-skewed and its median overstates the variance, so the MOR runs high; the overstatement shrinks as strata are added.

Usage

```
maihda_mor(model)
```

Arguments

`model`	A `maihda_model` from `fit_maihda` fitted with a `binomial` (lme4), `bernoulli` (brms), or `cumulative` (ordinal) family and a **logit** link.

Value

A single number (the MOR, ≥ 1), or `NA_real_` if the between-stratum variance is unavailable – which for a crossed-dimensions fit also covers the case where the stratum grid needed for the mixture cannot be resolved (fewer than two strata, absent dimension columns, or more than 12 dimensions).

References

Larsen, K., & Merlo, J. (2005). Appropriate assessment of neighborhood effects on individual health: integrating random and fixed effects in multilevel logistic regression. *American Journal of Epidemiology*, 161(1), 81-88.

See Also

`maihda_discriminatory_accuracy`

---

maihda_proportional_odds_test

Parametric-bootstrap proportional-odds test for a cumulative MAIHDA fit

---

Description

Tests the proportional-odds (parallel-lines) assumption of a fitted cumulative `clmm` MAIHDA model by calibrating the omnibus nominal-effects likelihood-ratio statistic against its own null distribution under the fitted model.

Usage

```
maihda_proportional_odds_test(object, n_sim = 199, seed = NULL)
```

Arguments

object	A maihda_model fitted with engine = "ordinal".
n_sim	Number of parametric-bootstrap replicates (default 199).
seed	Optional integer seed, for a reproducible bootstrap.

Details

The statistic is the omnibus nominal-effects likelihood-ratio statistic: the fixed-effect part of the model is refitted twice with ordinal::clm() – once with all covariates proportional, once with every covariate entering as a threshold-specific (nominal) effect – and twice the log-likelihood difference is taken.

Its usual chi-squared reference (ordinal::nominal_test()) is not valid here. After the stratum random intercept is integrated out, the marginal cumulative-logit slopes generally differ across thresholds once the stratum variance is non-zero. The chi-squared reference also treats the observations as independent, whereas observations that share a stratum share its random effect. A correctly specified model is therefore rejected too often, and more often as the sample grows.

The null distribution is instead simulated under the fitted clmm: each replicate redraws the stratum random effects from $N(0, \tau^2)$ at the fitted variance, forms the conditional category probabilities from the fitted thresholds and location predictor, redraws the response, and recomputes the statistic. The p-value is $(1 + \#\{T_b \geq T_{obs}\})/(1 + B)$ over the B replicates that refitted. n_sim sets only its Monte Carlo resolution: the smallest attainable value is $1/(B + 1)$.

This is a parametric-bootstrap approximation, not an exact test. The replicates are drawn at the fitted stratum variance, which is estimated from only as many units as there are strata, so the test is least reliable when the strata are few, whatever n_sim is.

The test is expensive – every replicate refits two clm() models – and is not run at fit time.

Value

An object of class maihda_po_test: a list with lrt, df, n_terms, p_value (the bootstrap p-value), p_chisq, n_sim (replicates that produced a usable statistic), n_failed, and null_lrt (the simulated null statistics). p_value is the one to report and the only one printed. p_chisq is the chi-squared p-value of the same statistic; it is not calibrated for a mixed model (see Details) and is returned for comparison only.

See Also

fit_maihda, maihda_cumulative

Examples

```
strata <- make_strata(maihda_sim_data, vars = c("gender", "race"))
d <- strata$data
d$y <- factor(cut(d$health_outcome, 3), labels = 1:3, ordered = TRUE)
```

```
m <- fit_maihda(y ~ age + (1 | stratum), data = d, family = "ordinal")
maihda_proportional_odds_test(m, n_sim = 99, seed = 1)
```

`maihda_sim_data` *Simulated Health Data for MAIHDA Use*

Description

A simulated dataset for demonstrating Multilevel Analysis of Individual Heterogeneity and Discriminatory Accuracy (MAIHDA).

Usage

```
maihda_sim_data
```

Format

A data frame with 500 rows and 7 variables:

id Unique participant identifier.

gender Gender of the participant.

race Simulated race/ethnicity category.

education Educational attainment level.

age Age in years, a continuous covariate.

health_outcome A continuous simulated health outcome.

binary_outcome A binary version of the health outcome.

Source

Simulated for the purpose of the MAIHDA package.

Examples

```
data(maihda_sim_data)
strata_result <- make_strata(maihda_sim_data, vars = c("gender", "race", "education"))
```

`maihda_sparse_data` *Sparse Intersectional Data for Bayesian MAIHDA*

Description

A simulated cross-sectional dataset in which many intersectional strata hold only a handful of individuals, built with a known intersectional interaction. With strata this small the split of the between-stratum variance into additive and interaction parts is weakly identified. An lme4 fit of the crossed-dimensions decomposition can be singular; a brms fit of it (`engine = "brms"`) gives the split a posterior median and a credible interval, both of which depend on the prior on the standard deviations.

Usage

```
maihda_sparse_data
```

Format

A data frame with 240 rows and 6 variables:

gender Strata dimension (Women/Men).

ethnicity Strata dimension (White/Black/Asian).

education Strata dimension (Low/High).

age_group Strata dimension (Young/Mid/Older).

y A continuous (Gaussian) outcome with residual SD 1. The 36 stratum effects have a variance of 0.31 (a VPC of 0.24), of which 41% is the intersectional interaction.

event A binary outcome (No/Yes), ~46% "Yes". On the latent scale its stratum effects have a variance of 1.32 (a VPC of 0.29), again 41% interaction.

`attr(maihda_sparse_data, "truth")` holds, for each outcome, the intercept and the 36 stratum effects with their additive, interaction and between-stratum variances, the interaction share and the VPC.

Details

Four dimensions form 36 intersectional strata with deliberately skewed sizes (median 6 individuals, 12 of 36 cells below 5, two singletons). The stratum effects were drawn with the interaction aimed at 40% of the between-stratum variance; the 36 effects drawn carry 41% on both outcomes (on the latent scale for `event`), by the mean-square split of the section Realised strata of `maihda`. The interaction share a crossed-dimensions fit reports is a ratio of variance parameters, and the four dimensions' variances, each estimated from two or three levels, add up to more than the spread of the main effects among the 36 strata: fitted to these stratum effects with 1,000 individuals in every stratum, lme4 puts the share at about 28% on `y` and 35% on `event`, the gap between the two being that of REML and maximum likelihood (the two sets of effects differ only in scale). In `vignette("bayesian_sparse_maihda")` both lme4 fits are singular through a dimension's variance, not the interaction's, and put the share at about 23% on `y` and 3% on `event`. The brms posterior medians are below 41% as well; under the priors the vignette sets on the standard deviations, their credible intervals run from near zero to about 41% or more.

Note

A purely illustrative dataset. The dimension labels are arbitrary and the interaction is constructed, not estimated from any real population – its only purpose is to make the sparse-cell behaviour of the ML and Bayesian estimators visible against a known answer.

Source

Simulated; see `data-raw/maihda_sparse_data.R`.

Examples

```
data(maihda_sparse_data)
attr(maihda_sparse_data, "truth")$gaussian$interaction_share  # 0.41

# lme4: the interaction share as a point estimate, from a fit that is singular here:
# m_lme4 <- maihda(y ~ 1 + (1 | gender:ethnicity:education:age_group),
#                  data = maihda_sparse_data, decomposition = "crossed-dimensions")
#
# brms: a posterior median and a credible interval, both resting on the prior:
# m_brms <- maihda(y ~ 1 + (1 | gender:ethnicity:education:age_group),
#                  data = maihda_sparse_data, decomposition = "crossed-dimensions",
#                  engine = "brms",
#                  prior = brms::set_prior("normal(0, 0.5)", class = "sd"))
# See vignette("bayesian_sparse_maihda").
```

---

maihda_table *Canonical MAIHDA results table and ranked-strata table*

Description

Assembles the two standard MAIHDA write-up deliverables from a fitted analysis in one call: (a) a **model-results table** contrasting the null and adjusted models (intercept, between-stratum variance and SD, VPC/ICC, the PCV, and – for a binary outcome – the AUC and Median Odds Ratio), and (b) a **ranked-strata table** ordering the intersectional strata by their predicted outcome, so the best- and worst-off strata can be read directly. It introduces no new estimator: the model-results table reuses the quantities from `summary()` (calling `summary()` itself for a bare `fit_maihda` model), and the ranked-strata table reuses the same stratum predictions as `plot(type = "predicted")`, so the table agrees exactly with `summary()` and `plot()`.

Usage

```
maihda_table(
  x,
  n_strata = 10L,
  scale = c("response", "link"),
  which = c("null", "adjusted"),
  digits = 3,
  ...
)
```

Arguments

x	A maihda_analysis from maihda (the usual input), or a single maihda_model from fit_maihda.
n_strata	Number of strata to show at each end (top and bottom) in the printed ranked-strata table. The returned $strata holds all strata (NULL for a longitudinal fit, whose strata are trajectories rather than single ranked values – see $strata in Value). Default 10.
scale	Scale for the predicted stratum values: "response" (default) or "link". For a cumulative (ordinal) model the response scale is the expected category score.
which	For a two-model analysis, which model's predictions to rank the strata by: "null" (default) or "adjusted". Ignored for a crossed-dimensions analysis or a single model.
digits	Number of decimal places for the print() method. Default 3.
...	For a maihda_model input, additional arguments passed to summary.maihda_model (e.g. bootstrap = TRUE); ignored for a maihda_analysis input, whose summaries are already computed.

Details

The model-results table is mostly numeric and export-ready (e.g. write.csv(maihda_table(a)$models, ...) or pass it to knitr::kable()): statistics are rows, models are columns, and each estimate has accompanying *_lower/*_upper columns that hold the confidence/credible interval when one is available (the intercept's interval, the VPC bootstrap or posterior interval, and the bootstrap PCV interval) and NA otherwise. The intercept carries the same interval summary() reports for it – a Wald interval for the likelihood engines, a credible interval for brms. The variance and SD rows are point estimates. The print() method renders the same table in the familiar "estimate [low, high]" layout.

For a "crossed-dimensions" analysis (one model, no null/adjusted pair) the results table has a single estimate column and gains "Additive share" / "Interaction share" rows instead of the PCV. For a contextual cross-classified analysis (maihda(context =)) it gains a "Context share (VPC)" row. A bare fit_maihda model is also accepted and yields a single-model table (no PCV).

The ranked-strata table ranks every stratum by its model-predicted outcome (on the scale requested), using the same stratum predictions as plot(type = "predicted"): the predicted value carries the conditional (random-effect) interval, and the stratum random effect (BLUP) is reported alongside it. By default the ranking uses the **null** model – the headline intersectional inequality (which strata fare best/worst overall); set which = "adjusted" to rank by the adjusted model instead. The full ranked table is returned in $strata; print() shows the top and bottom n_strata.

Value

An object of class maihda_table: a list with

models	a data frame of the model-results table (statistics in rows; one estimate column per model, each with *_lower/*_upper interval columns)
strata	a data frame of all strata ranked by predicted outcome, with rank, stratum, label, n, the predicted value and its conditional interval, and the stratum random effect and its interval. NULL when a single ranked value per stratum is not

defined – in particular for a longitudinal (growth-curve) fit, whose strata are trajectories (see `strata_note`) – or if the stratum predictions could not otherwise be computed

`strata_note` a character note explaining why `strata` is `NULL` (e.g. a longitudinal fit), or `NULL` when a ranked-strata table was produced

`models_note` a character note (or `NULL`) flagging that the PCV was computed from maximum-likelihood-refitted between-stratum variances while the variance/VPC rows are each model's own (REML) estimate – or from REML-type variances (`estimation = "REML"`) while those rows are maximum-likelihood estimates – so the PCV need not equal the reduction implied by the displayed variance rows; see `calculate_pcv`

`model_keys`, `model_labels`
the estimate-column keys and their display labels

`family`, `engine`, `mode`, `scale`, `ranked_by`, `n_obs`, `n_strata_total`, `context_vars`
metadata used by `print()`

See Also

`maihda`, `summary.maihda_model`, `calculate_pcv`, `maihda_discriminatory_accuracy`.

Examples

```
data(maihda_health_data)
a <- maihda(BMI ~ Age + Gender + Race + (1 | Gender:Race), data = maihda_health_data)

tab <- maihda_table(a)
tab                 # printed: model-results table + top/bottom strata
tab$models          # the numeric, export-ready results table
tab$strata          # all strata ranked by predicted BMI

# write.csv(tab$models, "results.csv", row.names = FALSE)
```

---

maihda_tidiers *Tidy a MAIHDA summary, model, or analysis*

Description

`tidy` methods that return the MAIHDA estimates as a tidy `tibble`, ready for downstream tables (`gt`, `flextable`) and `ggplot2`. They read the slots that `summary.maihda_model` already computes and add no new statistics.

Usage

```
## S3 method for class 'maihda_summary'
tidy(x, component = c("strata", "variance", "fixed"), ...)

## S3 method for class 'maihda_model'
tidy(x, component = c("strata", "variance", "fixed"), ...)

## S3 method for class 'maihda_analysis'
tidy(
  x,
  component = c("strata", "variance", "fixed"),
  which = c("null", "adjusted"),
  ...
)
```

Arguments

x	A `maihda_summary` (from `summary`), a `maihda_model` (from `fit_maihda`), or a `maihda_analysis` (from `maihda`).
component	Which estimates to return: `"strata"` (default) the stratum (intersection) random-effect estimates – one row per stratum, with `estimate`, `std.error` and `conf.low`/`conf.high`, plus the human-readable intersectional `label` when available. `"variance"` the variance-components table (between-stratum, any other random effects, residual, and total) with each component's variance, SD and proportion. `"fixed"` the fixed-effect estimates in broom's `term`/`estimate`/`std.error`/`statistic`/`p.value`/`conf.low`/`conf.high` shape.
...	Unused, for S3 consistency.
which	For a `maihda_analysis`, whether to tidy the `"null"` (default) or `"adjusted"` model's summary.

Value

A `tibble`. For `component = "strata"`: columns `stratum`, `label`, `estimate`, `std.error`, `conf.low`, `conf.high`. For `"variance"`: `component`, `variance`, `sd`, `proportion`. For `"fixed"`: `term`, `estimate`, `std.error`, `statistic`, `df`, `p.value`, `conf.low`, `conf.high`.

Fixed-effect statistics

For the lme4, WeMix and ordinal engines `statistic` is the Wald statistic (`estimate / std.error`), `p.value` its two-sided p-value, `conf.low`/`conf.high` the matching Wald interval at the summary's `conf_level`, and `df` the denominator degrees of freedom they were built on.

A Gaussian **lme4** fit reports containment (between-within) `df` and a t; every other engine leaves `df` `NA` and uses a z. Under `summary(df_method = "bootstrap")` `p.value` and the interval come from the parametric bootstrap and `df` is `NA`. See `summary.maihda_model`. For brms the estimate is the posterior mean with its `Est.Error` and credible interval, and `statistic`/`df`/`p.value` are `NA`. Standard errors are `NA` where they are undefined (a boundary fit).

See Also

glance.maihda_analysis for the one-row model summary.

Examples

```
data("maihda_health_data")
m <- fit_maihda(BMI ~ Age + (1 | Gender:Race:Education), data = maihda_health_data)
tidy(m)                       # stratum estimates
tidy(m, component = "variance")
tidy(m, component = "fixed")
```

---

maihda_upset_size *Recommended Figure Size for the UpSet Stratum Plot*

Description

Computes sensible width and height (in inches) for plot(object, type = "upset"), so a knitr chunk or ggsave() call can size the figure to its content. The UpSet composite grows *taller* with the number of matrix rows (one per binary 0/1 dimension, one per level of a multi-level factor) and *wider* with the number of strata columns shown, so a single fixed size tends to crop or stretch it – particularly for multi-level designs (many rows) or a large n_strata (many columns; UpSet is an inherently wide format).

Usage

```
maihda_upset_size(object, n_strata = 50)
```

Arguments

object	A maihda_model from fit_maihda or a maihda analysis from maihda; it must carry the per-dimension stratum table from make_strata.
n_strata	Maximum number of strata the plot will show – pass the same value you give plot(). NULL means all strata. Default 50.

Value

A list with numeric width and height (inches) plus the rows (matrix rows) and cols (strata shown) they derive from.

See Also

plot.maihda_model

Examples

```
strata <- make_strata(maihda_sim_data, vars = c("gender", "race"))
model <- fit_maihda(health_outcome ~ (1 | stratum), data = strata$data)
sz <- maihda_upset_size(model, n_strata = 30)
ggplot2::ggsave(
  tempfile(fileext = ".png"),
  plot(model, type = "upset", n_strata = 30),
  width = sz$width, height = sz$height)
```

`maihda_vpc_response` *Response-scale VPC for a binomial or count MAIHDA model*

Description

Computes the variance partition coefficient on the scale of the observed response: the probability scale for a binomial MAIHDA model, the count scale for a Poisson or negative-binomial one.

For a **binomial** model it uses the simulation method of Goldstein, Browne & Rasbash (2002). Stratum random effects $u \sim N(0, \sigma_u^2)$ are simulated and converted to predicted probabilities $p = g^{-1}(\eta + u)$ (with η the fixed-part linear predictor); the VPC is then the between-stratum variance of p as a share of the total (between + the binomial within-stratum variance $\overline{p(1-p)}$). Unlike the latent-scale VPC (fixed level-1 variance $\pi^2/3$ for the logit), the response-scale VPC depends on the overall outcome prevalence, so report it as a complement to – not a replacement for – the latent-scale value.

For a **Poisson or negative-binomial** model with the log link it evaluates the exact expressions of Leckie, Browne, Goldstein, Merlo & Austin (2020). With $\lambda = \exp(\eta + \sigma_u^2/2)$ the marginal expected count, the between-stratum variance of the counts is $\lambda^2(\exp(\sigma_u^2) - 1)$ and their within-stratum variance is $\lambda + \alpha\lambda^2 \exp(\sigma_u^2)$, where $\alpha = 1/\theta$ is the negative-binomial overdispersion (0 for a Poisson model); the VPC is the between-stratum variance as a share of their sum. It is the share of the variance of the counts themselves that lies between strata, and it involves no approximation of the level-1 variance. The latent-scale VPC of `summary.maihda_model` refers to the log scale instead and depends on the `count_approximation` of `fit_maihda`, so the two answer different questions. Report the count-scale VPC when the partition should describe the observed counts, or when the expected count is low and the latent-scale approximations disagree.

Usage

```
maihda_vpc_response(model, n_sim = 10000, seed = NULL)
```

Arguments

`model`	A binomial, Poisson or negative-binomial `maihda_model` (lme4 engine) from `fit_maihda`.
`n_sim`	Number of Monte-Carlo draws of the stratum random effect (>= 100) for a binomial model. Default 10000.
`seed`	Optional integer seed for the binomial simulation.

Details

Binomial models. The fixed part η is collapsed to a single value – the mean linear predictor $\bar{\eta}$ over the analytic sample – before the random effect is simulated around it. The result is therefore a VPC *evaluated at the mean covariate profile* (a conditional-at-mean estimate), not one marginalised over the empirical covariate distribution. For the canonical strata-only (null) model η is constant (the intercept), so the two coincide and the value is exact. For an *adjusted* model (one with covariates) they can differ, because the inverse link is nonlinear and $g^{-1}(\bar{\eta}) \neq \overline{g^{-1}(\eta)}$: read the response-scale VPC from the null model, or interpret an adjusted value as conditional on the average covariate profile rather than as a covariate-averaged (marginal) VPC.

The method is binomial-link agnostic: it maps the simulated stratum effects through whichever inverse link the model uses (logit, probit, cloglog, ...), so a non-logit binomial fit is computed on its own scale rather than rejected. Only the family is required to be binomial.

When the model carries random intercepts *beyond* the stratum (a contextual `(1 | school)` or an explicit `(1 | site)`), the simulation integrates over them: the reported estimate is the *stratum share* $Var(E[p \mid u_{stratum}])$ of the total response-scale variance, where the total includes the variation the other random effects induce in p plus the binomial within-variance. Simulating the stratum effect alone would overstate the stratum share. The inner integral over the combined non-stratum effect is evaluated by *deterministic Gauss-Hermite quadrature*, not an inner Monte-Carlo sample, so it contributes no simulation error of its own and increasing `n_sim` converges the whole estimate (the outer stratum draw is then the only Monte-Carlo dimension). The number of quadrature nodes used is reported in `inner_nodes`.

Count models. λ is computed for each observation of the analytic sample from its fixed-part linear predictor, any offset included, so with covariates or an offset the VPC differs between observations. The `estimate` is the mean of the observation-level VPCs; their median, quartiles and range are returned with it, as are the mean and range of λ. Prior `weights` of the fit enter these summaries as frequencies. A null model without an offset has a single λ and a single VPC. `n_sim` and `seed` are not used.

The expressions are those of a model whose only random effect is the stratum random intercept. A count model with a `context` or any other additional random effect, a crossed-dimensions fit, a longitudinal fit, and a link other than the log are refused.

Value

An object of class `maihda_vpc_response`: a list with `estimate`, `scale` = `"response"`, `method` (`"simulation"` for a binomial model, `"closed-form"` for a count model), `var_between` (the latent-scale between-stratum variance) and `var_other` (the summed latent-scale variance of any non-stratum random intercepts, 0 when there are none).

For a binomial model it also holds `n_sim`, `lp_fixed` (the mean fixed-part linear predictor), `inner_method` (`"gauss-hermite"` when non-stratum effects are integrated out, else `"none"`) and `inner_nodes` (the number of Gauss-Hermite quadrature nodes used for that inner integral, `NA` when there are no non-stratum effects).

For a count model it also holds `family`, the `median`, `quartiles` and `range` of the observation-level VPCs (all equal to `estimate` when λ is the same for every observation), `lambda` (the mean marginal expected count), `lambda_range` and `theta` (the negative-binomial dispersion, `Inf` for a Poisson model).

References

Goldstein, H., Browne, W., & Rasbash, J. (2002). Partitioning variation in multilevel models. *Understanding Statistics*, 1(4), 223-231.

Leckie, G., Browne, W. J., Goldstein, H., Merlo, J., & Austin, P. C. (2020). Partitioning variation in multilevel models for count data. *Psychological Methods*, 25(6), 787-801. doi:10.1037/met0000265

See Also

maihda_discriminatory_accuracy, summary.maihda_model

Examples

```
strata <- make_strata(maihda_health_data, vars = c("Gender", "Race"))
d <- maihda_health_data
d$stratum <- strata$data$stratum
m <- fit_maihda(Obese ~ (1 | stratum), data = d, family = "binomial")
maihda_vpc_response(m, seed = 1)

# A count outcome: the VPC on the count scale, in closed form
set.seed(1)
u <- rnorm(max(d$stratum), sd = 0.4)
d$visits <- rpois(nrow(d), exp(0.5 + u[d$stratum]))
m_count <- fit_maihda(visits ~ (1 | stratum), data = d, family = "poisson")
maihda_vpc_response(m_count)
```

---

make_strata *Create Strata from Multiple Variables*

Description

This function creates strata (intersectional categories) from multiple categorical variables in a dataset.

Usage

```
make_strata(
  data,
  vars,
  sep = " × ",
  min_n = 1,
  autobin = TRUE,
  bin_rows = NULL
)
```

Arguments

data	A data frame containing the variables to create strata from.
vars	Character vector of variable names to use for creating strata.
sep	Separator to use between variable values when creating stratum labels. Default is " \u00d7 " (a mathematical multiplication sign).
min_n	Minimum number of observations required for a stratum to be included. Strata with fewer observations will be coded as NA. Default is 1.
autobin	Logical indicating whether to automatically bin numeric grouping variables with more than 10 unique values into 3 categories (tertiles). Default is TRUE. When this happens a `message()` is emitted, because the resulting strata are data-dependent (tertile cut-points depend on the sample) and a continuous variable placed in the grouping term is usually unintended. Set `autobin = FALSE` to disable, or bin the variable yourself for explicit, reproducible cut-points.
bin_rows	Optional logical vector, one element per row of `data`, selecting the rows used to compute the auto-bin cut-points (every row is still assigned a stratum). Mainly for internal use by `fit_maihda`, which passes the `subset` rows so that `fit_maihda(..., subset = keep)` bins on the same sample as fitting `data[keep, ]`. Default `NULL` uses all rows.

Details

If any of the specified variables has a missing value (NA) for a given observation, that observation will be assigned to the NA stratum (stratum = NA), rather than creating a stratum that includes the missing value.

The strata_info data frame is also attached as an attribute to the data, which allows fit_maihda() to automatically capture stratum labels for use in plots and summaries.

When `autobin` discretises a numeric grouping variable `v`, the adjusted-model and prediction machinery later add an internal factor column named `.maihda_dim_<v>`; the `.maihda_dim_` prefix is therefore reserved. `make_strata()` errors if `data` already holds the `.maihda_dim_<v>` column for a variable it is about to auto-bin, so an existing user column is never silently overwritten (rename it, or pass `autobin = FALSE`).

A numeric grouping variable with 10 or fewer unique values (or any numeric when `autobin = FALSE`) is kept as its raw values: distinct values still define distinct strata, but the variable is *not* binned. If such a variable is then used to build an adjusted MAIHDA (`maihda`), it enters the adjusted model as a linear fixed effect rather than categorical main effects, and `maihda()` warns (when it has three or more distinct values). Wrap category codes in `factor()` before creating strata if you want them treated categorically.

Value

An object of class `maihda_strata`: a list with elements

data	The original data frame with an added 'stratum' column. The strata_info is also attached as an attribute for use by fit_maihda()
strata_info	A data frame with information about each stratum including counts and the combination of variable values

vars	The stratum-defining variable names, as supplied.
sep	The label separator used.
min_n	The minimum stratum size applied.
autobin_info	A named list with the breaks and labels of each auto-binned numeric variable (empty when nothing was binned).

Examples

```
# Create strata from gender and race variables
result <- make_strata(maihda_sim_data, vars = c("gender", "race"))
print(result$strata_info)
```

---

pcv_importance *Order-Invariant PCV Attribution Across Predictors (Shapley / Dominance)*

Description

Apportions the between-stratum variance reduction (the PCV) among a set of predictors. Where stepwise_pcv adds the variables one at a time – so each variable's contribution depends on its entry order – pcv_importance() treats the PCV as a value function over variable subsets and attributes it *fairly*: the flagship "shapley" method averages each variable's marginal PCV over every possible entry order, and "dominance" adds Budescu's pairwise dominance detail (its general dominance weights coincide with the Shapley values). Both methods satisfy the **efficiency** identity: the contributions sum exactly to the full model's total PCV.

Two attribution targets are useful in a MAIHDA. Passing the *stratum dimensions* (e.g. c("gender", "race", "education")) splits the additive share – the canonical null-to-adjusted PCV – fairly across the dimensions: which social dimension drives the additive between-stratum inequality. Passing *individual-level covariates* gives an order-invariant counterpart to stepwise_pcv(): how much each covariate explains of the between-stratum variance (subject to the latent-scale caveat below for non-Gaussian families).

Usage

```
pcv_importance(
  data,
  outcome,
  vars,
  method = c("shapley", "dominance"),
  approx = c("exact", "montecarlo"),
  n_perm = 2000,
  engine = "lme4",
  family = "gaussian",
  context = NULL,
  sampling_weights = NULL,
```

```
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  estimation = c("fitted", "ML")
)
```

Arguments

data	Data frame with observations. Ensure make_strata was run first so the stratum variable exists.
outcome	Character string; the dependent variable.
vars	Character vector of predictors (stratum dimensions and/or covariates) among which the PCV is apportioned. Their order does not affect the exact attribution; a Monte-Carlo attribution depends on it only through its sampling error.
method	Attribution method: "shapley" (default; order-invariant Shapley values) or "dominance" (Budescu dominance analysis: general dominance – equal to the Shapley values – plus the conditional and complete dominance detail).
approx	For method = "shapley" only: "exact" fits all $2^k - 1$ non-empty variable subsets (plus the null); "montecarlo" samples n_perm random entry orders instead. When approx is not supplied it defaults to "exact" for up to 10 variables and switches to "montecarlo" (with a message) beyond. "dominance" requires every subset, so it errors if approx = "montecarlo" is requested.
n_perm	Number of random permutations for approx = "montecarlo". Default 2000. Distinct subset fits are cached, so the number of *model fits* is bounded by min(2^k, n_perm * (k - 1) + k + 1); set a seed (set.seed()) for reproducible sampling.
engine	Modeling engine ("lme4", "brms", "wemix", or "ordinal"); default "lme4". Resolved exactly as in stepwise_pcv: switches to "wemix" when sampling_weights is supplied and to "ordinal" for an ordinal family or ordered-factor outcome. Exact attribution with engine = "brms" refits a Stan model for *every* subset and is strongly discouraged beyond a handful of variables (see Details).
family	Error distribution and link function. Default "gaussian"; a binary or ordered-factor outcome is auto-detected when family is left unspecified, mirroring stepwise_pcv.
context	Optional higher-level context column(s) (e.g. "school"), forwarded to every subset fit so each model carries the crossed (1 | context) intercept alongside the stratum effect; the attribution is then of the between-stratum PCV **net of** the context, exactly as in stepwise_pcv. lme4/brms engines only.
sampling_weights	Optional name of a sampling-weight column for design-weighted fits; see fit_maihda. The weight column joins the complete-case filter so every subset fit uses the same analytic sample.
bootstrap	Logical; compute parametric-bootstrap confidence intervals for each contribution by refitting *every* subset model on responses simulated from the full model. **lme4 engine and exact attribution only**; the cost is n_boot times the number of subset models (up to n_boot * 2^k refits). Default FALSE.
n_boot	Number of bootstrap draws if bootstrap = TRUE. Default 1000.

`conf_level` Confidence level for bootstrap intervals. Default 0.95.

`estimation` Variance-estimation basis for the between-stratum variances the attribution differences across subset models, `"fitted"` (default) or `"ML"`; see `calculate_pcv`. Affects Gaussian `lmer` fits only.

Details

Value function and efficiency. Write V_0 for the null model's between-stratum variance and $V(S)$ for the between-stratum variance after adding the variable subset S as fixed effects (each put on the `estimation` basis – the default `"fitted"` keeps each `lmer` fit's own REML variance, `"ML"` refits it with maximum likelihood – exactly as in `calculate_pcv` and `stepwise_pcv`, so all attributions live on the same scale as the rest of the package). The value function is $v(S) = (V_0 - V(S))/V_0$ – the total PCV of the model that adds S – and the Shapley contribution of variable i averages its marginal PCV $v(S \cup \{i\}) - v(S)$ over all subsets with the usual Shapley weights. Because every permutation's marginals telescope to $v(N)$, the contributions of *every* method here (including the Monte-Carlo approximation, for any `n_perm`) sum exactly to the full-model total PCV. This is the multilevel-PCV analogue of the LMG / Shorrocks-Shapley decomposition of R^2 (Groemping 2006; Shorrocks 2013).

Exact vs. Monte-Carlo cost. The exact attribution fits $2^k - 1$ non-empty subset models plus the null – 2^k fits in total, with every fit cached and reused across all marginal differences: 256 fits at $k = 8$, 1024 at $k = 10$. That is feasible for `lme4` but quickly infeasible beyond, and *each* of those fits is a separate Stan run under `engine = "brms"` – exact attribution on brms is therefore strongly discouraged except for very small k. The Monte-Carlo route samples `n_perm` entry orders (unbiased for the Shapley values, since a uniform random permutation reproduces the Shapley weights), reports a per-variable Monte-Carlo standard error, and warns when the largest `MC_SE` exceeds 0.01 on the PCV scale (increase `n_perm`).

Latent-scale families and rescaling. For binomial/ordinal (and, in attenuated form, count) families, adding a predictor that varies *within* strata rescales the latent metric, so its marginal PCV mixes explained variance with rescaling (see the latent-scale note in `calculate_pcv`); Shapley values of the PCV inherit this. Attribution among the stratum *dimensions* – constant within each stratum – is largely unaffected.

Suppression and negative contributions. The PCV can be negative, so a contribution (and its `Share`) can be negative or exceed 100%. Efficiency still holds; contributions are reported *signed* and no non-negative normalisation is applied. A negative Shapley contribution flags a suppressor-style variable whose inclusion tends to *raise* the between-stratum variance.

Bootstrap. With `bootstrap = TRUE` the whole attribution is bootstrapped: responses are simulated from the full model and every subset model is refit per draw (`lme4::refit`), giving percentile intervals per contribution – `n_boot` times `n_fits` refits, so gate it by cost. Available for `engine = "lme4"` with exact attribution (any `method`); the Monte-Carlo approximation already carries its own sampling error and is not combined with the bootstrap. As in `calculate_pcv`, `refit()` holds a `glmer.nb` dispersion parameter fixed at its original estimate.

Value

An object of class `maihda_pcv_importance`: a list with

`importance` Data frame with one row per variable (in the order of `vars`): `Contribution` (the variable's share of the total PCV, on the PCV scale), `Share` (`Contribution` /

total_pcv; NA when the total PCV is zero), plus MC_SE (Monte-Carlo standard errors, approx = "montecarlo" only) and CI_lower/CI_upper (bootstrap = TRUE only).

total_pcv
: The full-model PCV, $(V_0 - V_{full})/V_0$; the contributions sum to this value (efficiency).

null_variance, full_variance
: Between-stratum variance of the null and the full (all-vars) model.

subsets
: Data frame of every subset model fit: the variables in the subset, its size, between-stratum variance, and PCV $v(S)$.

conditional, complete_dominance
: method = "dominance" only: the conditional dominance matrix (variables x adjustment-set size) and the pairwise complete-dominance matrix.

method, approx, n_perm, n_fits, n_obs, engine, family, context, bootstrap, conf_level, n_boot_ok, n_boot_boundary, n_boot_nonconverged, interval_reliable, estimation, estimation_used, ml_variances
: Metadata; n_fits counts the distinct models fit (including the null), n_boot_ok the number of successful bootstrap draws, n_boot_boundary the draws excluded because the null between-stratum variance hit the zero boundary (the intervals are then conditional on a positive null variance), n_boot_nonconverged the retained draws whose refit optimiser did not converge, interval_reliable is FALSE when more than half the contributing draws did not converge (the intervals are still returned but flagged unreliable), estimation is the variance-estimation basis ("fitted"/"ML") requested for every subset model's between-stratum variance, estimation_used is the basis actually used ("mixed" when an ML refit was skipped at the boundary, leaving a subset model on REML), and ml_variances is TRUE when the compared variances are maximum-likelihood estimates (glmer, wemix or ordinal fits, or lmer fits refitted by maximum likelihood), which print() notes under the default estimation, as calculate_pcv does.

Reproducibility

approx = "montecarlo" draws random permutations with the session RNG; call set.seed() first for reproducible results. The exact methods give the same contributions on every run of a likelihood engine, but the bootstrap = TRUE intervals come from simulated responses and an engine = "brms" fit is an MCMC draw, so call set.seed() first for those as well.

References

Budescu, D. V. (1993). Dominance analysis: a new approach to the problem of relative importance of predictors in multiple regression. *Psychological Bulletin*, 114(3), 542-551.

Groemping, U. (2006). Relative importance for linear regression in R: the package relaimpo. *Journal of Statistical Software*, 17(1), 1-27.

Shorrocks, A. F. (2013). Decomposition procedures for distributional analysis: a unified framework based on the Shapley value. *Journal of Economic Inequality*, 11, 99-126.

See Also

stepwise_pcv for the sequential path with the discriminatory-accuracy trajectory, calculate_pcv for the two-model PCV and the latent-scale caveat, and maihda for the canonical null-vs-adjusted decomposition.

Examples

```
strata <- make_strata(maihda_sim_data, c("gender", "race"))

# Order-invariant split of the PCV across the two dimensions and age
imp <- pcv_importance(strata$data, "health_outcome", c("gender", "race", "age"))
print(imp)
plot(imp)

# Dominance analysis: adds conditional / complete dominance detail
pcv_importance(strata$data, "health_outcome", c("gender", "race", "age"),
               method = "dominance")

# Monte-Carlo approximation (set a seed for reproducibility)
set.seed(42)
pcv_importance(strata$data, "health_outcome", c("gender", "race", "age"),
               approx = "montecarlo", n_perm = 500)
```

---

plot.maihda_analysis *Plot a MAIHDA Analysis*

Description

Dispatches each type to the model it is valid on. The VPC and shrinkage views ("vpc", "obs_vs_shrunken", "predicted") use the **null** model by default; for type = "vpc" the model argument selects the null model, the adjusted model, or "both" – one combined plot that shows the null-to-adjusted change with the PCV annotated. The additive-vs-intersectional views ("effect_decomp", "prediction_deviation") use the **adjusted** model, whose fixed effects carry the dimensions' additive part so the stratum random effect is the pure interaction; with fewer than two dimensions (no adjusted model) they fall back to the null model. Group types ("group_vpc", "group_components", "group_between_variance", "group_pcv", "group_additive_share") use the group comparison when maihda was called with a group.

Usage

```
## S3 method for class 'maihda_analysis'
plot(
  x,
  type = "all",
  highlight_interactions = FALSE,
  only_flagged = FALSE,
```

```
  highlight_by = c("flag", "rope"),
  rope = NULL,
  select = c("order", "deviation"),
 order_by = c("predicted_desc", "stratum", "predicted_asc", "deviation", "size"),
  model = c("null", "adjusted", "both"),
  ...
)
```

Arguments

x	A maihda_analysis object from maihda.
type	One of the model types ("all", "vpc", "obs_vs_shrunken", "predicted", "upset" (the UpSet-style alternative to "predicted"; forwards quantity via ...), "effect_decomp", "prediction_deviation"), the contextual type ("context_vpc", a stratum-vs-context variance bar; requires maihda(context =)), a longitudinal type ("vpc_trajectory", "trajectories", "pcv_trajectory"; requires decomposition = "longitudinal"), or a group type ("group_vpc", "group_components", "group_between_variance", "group_pcv", "group_additive_share"). Default "all". For a longitudinal analysis "all" shows the VPC-over-time, the stratum trajectories, and the time-specific PCV.
highlight_interactions	Highlight strata with a credibly non-zero intersectional interaction on the BLUP-based views (see maihda_interactions and plot). FALSE (default), TRUE (computed from this analysis's adjusted / crossed-dimensions model), a multiple-testing method such as "BH", or a maihda_interactions object. The flags are computed once from the correct (adjusted) model and reused across views.
only_flagged	Show only the highlighted strata on the "predicted" and "obs_vs_shrunken" views instead of dimming the rest (see plot). FALSE (default) keeps every stratum; TRUE restricts those views to the highlighted strata (those carrying a credibly non-zero interaction, or – under highlight_by = "rope" – those classified ROPE-"relevant") and, when highlight_interactions is left FALSE, turns the highlight on with this analysis's stored diagnostic. "effect_decomp" ignores it (it stays highlighted in context).
highlight_by	Which interaction-screen column defines the highlighted strata: "flag" (default, the zero-centred flagged column) or "rope" (the equivalence decision column, highlighting the strata classified "relevant"). See plot. "rope" requires rope (or a highlight_interactions object built with one).
rope	Equivalence region forwarded to maihda_interactions when computing the screen, so highlight_by = "rope" has a decision column to read: a single positive d for the symmetric region c(-d, d) on the latent (link) scale, or c(lower, upper). NULL (default) adds no equivalence classification.
select	When the n_strata cap drops strata on the "predicted" (or longitudinal "trajectories") view, which to keep: "order" (default, first n_strata in stratum order) or "deviation" (the n_strata furthest from the reference, both tails). See plot.
order_by	Display order of the strata on the "predicted" and "upset" views (display-only; does not change which strata are shown, nor any value): "predicted_desc" highest predicted first, "stratum" native order, "predicted_asc" lowest first,

	`"deviation"` largest `|predicted - reference|` first, or `"size"` largest stratum first. The default is per view – `"predicted_desc"` for `"predicted"`, `"size"` for `"upset"`. See `plot`.
`model`	For `type = "vpc"`, which model's variance partition to show: `"null"` (default) the total between-stratum heterogeneity, `"adjusted"` the between-stratum heterogeneity remaining after the dimensions' additive main effects (closer to the pure intersectional component), or `"both"` – a **single** plot placing the two partitions together and annotating the PCV, so the null-to-adjusted change is visible in one figure. The null and adjusted single views are labelled with a `"Null model"` / `"Adjusted model"` subtitle. For a **longitudinal** analysis the VPC view is the time-varying VPC trajectory, and `"both"` overlays the null and adjusted curves. A `"crossed-dimensions"` analysis fits a single model (no null/adjusted pair), so only `"null"` is valid there; `"adjusted"`/`"both"` error. `model` applies to the VPC view only – combining a non-default `model` with another `type` is an error.
`...`	Additional arguments passed to the underlying plot method.

Value

A ggplot2 object – including `type = "vpc"`, `model = "both"`, which is a single combined change plot – or (for `type = "all"`) an invisible list of them.

`plot.maihda_comparison`

Plot a MAIHDA Model Comparison

Description

Plots VPC/ICC across the models compared by `compare_maihda`. Dispatched via `plot()` on the classed result.

Usage

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

Arguments

`x`	A `maihda_comparison` object from `compare_maihda`.
`...`	Additional arguments (not used).

Value

A ggplot2 object.

Examples

```
strata <- make_strata(maihda_sim_data, vars = c("gender", "race"))

null_model <- fit_maihda(health_outcome ~ 1 + (1 | stratum), data = strata$data)
adj_model  <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata$data)

comparison <- compare_maihda(null_model, adj_model, bootstrap = TRUE)
plot(comparison)
```

---

plot.maihda_describe *Plot a MAIHDA sample description*

Description

Descriptive plots for a maihda_describe object: the stratum-size distribution (the sparsity of the strata space), the family-aware outcome distribution, and per-variable missingness.

Usage

```
## S3 method for class 'maihda_describe'
plot(x, type = c("stratum_size", "outcome", "missingness"), ...)
```

Arguments

x	A maihda_describe object.
type	One of "stratum_size" (default; histogram of stratum sizes with the small-stratum threshold marked, empty strata included as zeros when they were enumerated), "outcome" (histogram of a continuous/count outcome, or category bars for a binomial/ordinal one), or "missingness" (percent missing per variable).
...	Additional arguments (not used).

Value

A **ggplot2** object, extendable with the usual + grammar.

Examples

```
desc <- maihda_describe(BMI ~ Age + (1 | Gender:Race:Education),
                        data = maihda_health_data)
plot(desc, type = "stratum_size")
plot(desc, type = "outcome")
plot(desc, type = "missingness")
```

plot.maihda_group_comparison *Plot a MAIHDA Group Comparison*

Description

Visualises the output of compare_maihda_groups as a point/forest plot of the VPC/ICC by group, as stacked variance-composition bars (between- vs within-stratum share) by group, as bars of the absolute between-stratum (intersectional) variance by group, or as bars of the additive share (PCV) by group. Dispatched via plot() on the classed result.

Usage

```
## S3 method for class 'maihda_group_comparison'
plot(
  x,
  type = c("vpc", "components", "between_variance", "pcv", "additive_share"),
  ...
)
```

Arguments

x	A maihda_group_comparison object from compare_maihda_groups.
type	One of "vpc" (default) for VPC by group with optional bootstrap confidence intervals, "components" for stacked variance proportions (additive / interaction / residual for a crossed-dimensions comparison, between / other / residual otherwise, with a separate context slice for a contextual comparison), "between_variance" for the absolute between-stratum variance by group, "pcv" for the two-model additive share (null -> adjusted proportional change in between-stratum variance) by group, or "additive_share" for the crossed-dimensions additive share by group. The VPC is a *share* of the unexplained variance; "between_variance" shows the *magnitude* the ratio cannot convey (two groups with very different VPCs can share a between-stratum variance, and vice versa); "pcv" requires strata defined by at least two dimensions.
...	Additional arguments (not used).

Value

A ggplot2 object.

Examples

```
data(maihda_health_data)
cmp <- compare_maihda_groups(BMI ~ Age + (1 | Gender:Race),
                             data = maihda_health_data, group = "Education")
plot(cmp, type = "vpc")
plot(cmp, type = "components")
```

```
plot(cmp, type = "between_variance")
```

`plot.maihda_model` *Plot MAIHDA Model Results*

Description

Creates various plots for visualizing MAIHDA model results including variance partition coefficient comparisons, observed vs. shrunken estimates, and predicted subgroup values with conditional intervals.

Usage

```
## S3 method for class 'maihda_model'
plot(
  x,
  type = c("all", "vpc", "obs_vs_shrunken", "predicted", "upset", "effect_decomp",
    "prediction_deviation", "context_vpc", "vpc_trajectory", "trajectories"),
  summary_obj = NULL,
  n_strata = 50,
  highlight_interactions = FALSE,
  only_flagged = FALSE,
  highlight_by = c("flag", "rope"),
  rope = NULL,
  select = c("order", "deviation"),
  order_by = c("predicted_desc", "stratum", "predicted_asc", "deviation", "size"),
  quantity = c("predicted", "interaction"),
  ...
)
```

Arguments

`x`	A maihda_model object from `fit_maihda()`.
`type`	Character string specifying plot type: • "vpc": Variance partition coefficient visualization • "obs_vs_shrunken": Observed vs. shrunken stratum means. The y-axis (model-based estimate) includes the fixed effects, so for a covariate-adjusted model the distance from the diagonal reflects both shrinkage *and* covariate adjustment, not shrinkage alone; it is a pure shrinkage view only for an intercept-only (null) model • "predicted": Predicted values for each stratum with **conditional** (random-effect) intervals: the stratum random effect's own interval – a Wald interval on the conditional SD of the BLUP for the likelihood engines, the posterior interval of u_j for brms – shifted onto the outcome scale by the fixed-effect prediction, which is held at its point estimate. They answer "does this

stratum depart from the dashed across-strata reference?", which is what a caterpillar plot is read for; they are *not* intervals for the predicted value itself. Holding the fixed effects at a point estimate stops the fixed-effect uncertainty carried inside the random-effect interval from cancelling against the random effect it is correlated with, so these intervals are *wider* than a full interval for the stratum's predicted outcome, markedly so when there are few strata. By default the strata are ordered highest-to-lowest predicted value (a ranked caterpillar plot); see `order_by` to change or disable the ordering

- "upset": UpSet-style alternative to `"predicted"` – an intersection-size bar, a category matrix encoding each stratum's level on every dimension, and the predicted-value panel, all sharing one column order. Replaces the long intersectional axis labels with the matrix. Binary 0/1 dimensions show as a single present/absent row; multi-level factors get one row per level. Columns are ordered by intersection size (largest first) by default; `order_by = "predicted_desc"` gives the ranked caterpillar of `"predicted"` with the matrix in place of the text labels
- "effect_decomp": Visualizes additive vs intersectional deviation from global mean. Only the stratum random effect and, for a crossed-dimensions fit, the dimension random effects enter; a contextual random effect, or any other grouping, belongs to neither component and is left out of the deviations and the global mean, as it is from the `"predicted"` values
- "prediction_deviation": Detailed deviation panels for individuals or strata
- "context_vpc": Stratum vs. context variance bars for a contextual cross-classified fit (`fit_maihda(context = )`); errors otherwise. For a crossed-dimensions fit with a context the between-stratum variance is drawn as its additive dimension and interaction bars
- "vpc_trajectory": Time-varying VPC/ICC curve for a **longitudinal** fit (`fit_maihda(id =, time =)`); errors otherwise. For a longitudinal model `"vpc"` and `"all"` also route here
- "trajectories": Predicted per-stratum mean trajectories over time (longitudinal fits only)
- "all": Generate all available plots (default if not specified)

`summary_obj` Optional maihda_summary object from `summary()`. If NULL, will be computed.

`n_strata` Maximum number of strata to display in the predicted plot. When there are more strata than this, the first `n_strata` (in stratum order) are shown and the plot caption notes how many were omitted. Default is 50. Use NULL for all strata.

`highlight_interactions`
Highlight the strata that carry a credibly non-zero intersectional interaction (from `maihda_interactions`) on the BLUP-based views (`"effect_decomp"`, `"predicted"`, `"obs_vs_shrunken"`); other views ignore it. `FALSE` (default) off; `TRUE` computes the flags with `maihda_interactions()` defaults; or pass a multiple-testing method such as `"BH"` or a `maihda_interactions` object to reuse a specific `conf_level`/ `adjust`. For the pure-interaction reading the model should be the adjusted (or crossed-dimensions) model – e.g. via `plot()` on a `maihda` analysis,

which routes these views to the adjusted model automatically. Which column of the screen drives the highlight set is governed by `highlight_by`.

`only_flagged` Show *only* the flagged strata rather than dimming the rest. `FALSE` (default) keeps every stratum (flagged ones highlighted); `TRUE` restricts the `"predicted"` and `"obs_vs_shrunken"` views to the highlighted strata (those carrying a credibly non-zero interaction, or – under `highlight_by = "rope"` – those classified ROPE-`"relevant"`), so a highlighted stratum is never hidden by the `n_strata` cap. When `TRUE` and `highlight_interactions` is left `FALSE`, the flags are computed with `maihda_interactions` defaults; pass `highlight_interactions` to choose the `conf_level`/`adjust`. A captioned empty panel is returned when no stratum is flagged. It does not apply to `"effect_decomp"` (whose waterfall exists to show each flagged stratum's place in the *full* distribution); that view stays highlighted. Independently of this argument, whenever interactions are highlighted the `n_strata` cap on `"predicted"` becomes flag-aware: every flagged stratum is kept and the remaining slots are filled according to `select`.

`highlight_by` Which interaction-screen column defines the highlighted strata: `"flag"` (default) uses the zero-centred `flagged` column (credibly non-zero interaction); `"rope"` uses the equivalence `decision` column, highlighting the strata classified `"relevant"` (interaction interval entirely outside the region of practical equivalence). `"rope"` requires a screen carrying a `decision` column: either pass `rope`, or supply a `maihda_interactions` object built with `rope`; otherwise it errors.

`rope` Equivalence region (a "smallest interaction of interest") forwarded to `maihda_interactions` when the screen is computed here (i.e. when `highlight_interactions` is `TRUE` or a `p.adjust` method name). `NULL` (default) adds no equivalence classification; a single positive `d` means the symmetric region `c(-d, d)` on the latent (link) scale, or supply `c(lower, upper)`. Needed for `highlight_by = "rope"` unless the supplied `maihda_interactions` object already carries a `decision` column. Ignored when a precomputed `maihda_interactions` object is passed (its own `rope` is used).

`select` When the `n_strata` cap must drop strata, which to keep: `"order"` (default; the first n_strata in stratum order) or `"deviation"` (the n_strata furthest from the reference line – largest `|predicted - reference|`, so the most extreme strata in *both* directions). Applies to `"predicted"` and, for a longitudinal fit, `"trajectories"` (where it keeps the strata whose trajectories swing furthest from the population curve). Flagged strata are always kept; this governs the fill and the unflagged case. `select` changes *which* strata appear; their left-to-right display order is a separate choice governed by `order_by`.

`order_by` For `type = "predicted"` and `type = "upset"`, how to order the strata that are displayed (**display-only** – it does not change *which* strata are shown, that is `n_strata`/`select`, nor the predicted values, intervals, reference line, or highlighted set): `"predicted_desc"` orders from the highest predicted value to the lowest, `"stratum"` keeps the native stratum order, `"predicted_asc"` orders from lowest to highest, `"deviation"` orders by largest absolute deviation from the reference line (`|predicted - reference|`), and `"size"` orders by intersection (stratum) size, largest first. The default is **per view**: `"predicted_desc"` for `"predicted"` (labels run from the highest predicted value down) and `"size"` for `"upset"` (the UpSet convention, which also makes its intersection-size bar

monotone). On the "upset" view the three value-based orders sort on the quantity the bottom panel actually shows, so they follow quantity – with quantity = "interaction" they order by the stratum random effect and "deviation" measures the distance from zero. Combining order_by = "predicted_desc" with type = "upset" gives the ranked caterpillar of the "predicted" view drawn against the UpSet category matrix instead of long text labels. Ignored by the other plot types.

quantity — For type = "upset", which quantity the bottom panel shows: "predicted" (default) the stratum's predicted value (fixed + random effect) against the across-strata reference line, or "interaction" the stratum random effect (the BLUP) against zero – the deviation from the model's fixed prediction, which is the *pure* intersectional interaction when the dimension main effects are in the model (the adjusted model). Ignored by the other plot types.

... — Additional arguments (not currently used).

Value

For a single type, a **ggplot2** object that you can extend with the usual + grammar (themes, labs(), added layers, or a replacement fill/colour scale). Some types return a richer object: "prediction_deviation" returns a **patchwork** of two panels and "upset" a **patchwork** of three panels (theme every panel at once with & theme_*()). type = "all" returns a named list of ggplot2 objects.

Examples

```
strata_result <- make_strata(maihda_sim_data, vars = c("gender", "race"))
model <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata_result$data)

# VPC plot
plot(model, type = "vpc")

# Single-type plots are ggplot objects -- restyle them with ggplot2:
plot(model, type = "vpc") +
  ggplot2::theme_classic() +
  ggplot2::labs(title = "Variance partition, restyled")

# Generate all plots (a named list); pick one out to restyle it:
plots <- plot(model)
plots$predicted + ggplot2::theme_bw()
```

---

plot.maihda_pcv_importance

Plot a PCV importance attribution

Description

Bar chart of the per-variable contributions to the total PCV, ordered by size, with a dashed zero line (negative bars flag suppression) and, when available, parametric-bootstrap confidence intervals. The bars sum to the full-model Total PCV (the efficiency property).

Usage

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

Arguments

`x`	A `maihda_pcv_importance` object from `pcv_importance`.
`...`	Additional arguments (not used).

Value

A ggplot object.

`plot_prediction_deviation_panels`

Plot Prediction Deviation Panels

Description

Creates an advanced, publication-ready two-panel dashboard for visualizing predicted values and highlighting the most notable cases or strata. What "notable" means depends on the model type, and the labelled points are *not* statistical outliers in the regression-diagnostic sense:

- Gaussian and Poisson (and the ordinal `"expected_score"` mode): the cases/strata whose prediction sits furthest from the mean prediction (largest deviation), ranked by absolute deviation.
- Binomial: the cases/strata with the largest absolute deviance residual, i.e. where the observed 0/1 outcome – for an aggregated binomial (`cbind(successes, failures)`, a proportion with its trial counts as prior weights, or a brms `y | trials(n)` fit), the observed successes out of trials – is least consistent with the fitted probability (worst-fit points), ranked by $|devianceresidual|$. As in R's binomial family, a prior weight counts as that many trials, so a 0/1 outcome with weights is scored as that many successes or failures. Every row's outcome is coded as the model coded it – by label, whatever the order of a factor's levels in `data` – and a factor with more than two levels, as in R's binomial family, as its first level against all the others. Predictions are per-trial probabilities for every binomial fit. A row whose outcome or trial count is unknown has no residual: it is left out of the stratum residual means, is never labelled, and is drawn as a hollow point.
- Ordinal `"surprise"` mode: the cases/strata with the highest surprise $-\log P(\text{observed category})$, i.e. the least probable observations under the model. It needs the observed response in `data`: a row whose response is missing is left out, and one whose category is not among the model's fitted categories is left out with a warning.

Usage

```
plot_prediction_deviation_panels(
  model,
  data = NULL,
  type = c("auto", "gaussian", "poisson", "binomial", "ordinal"),
  ordinal_mode = c("surprise", "expected_score"),
  top_n_labels = 5,
  strata_info = NULL
)
```

Arguments

model	A fitted model object (e.g., from `lm()`, `glm()`, `MASS::polr()`, or `lme4::glmer()`).
data	The original data frame used to fit the model. If `NULL`, attempts to extract from the model. When supplied, everything is computed for its own rows: their predictions, their binomial residuals (from their own outcomes and trial counts) and their weights in the stratum summaries. A row's weight is read from its column in `data` – the sampling-weight column, or the column a bare model's `weights` = argument names (`glm()`, `lmer()`, `MASS::polr()`, ...) – or else from the fitted row with the same row name and prediction (a `fit_maihda()` model stores its weights as values). If any plotted row's weight cannot be found, every row is weighted equally, with a warning.
type	Model type: "auto" (default), "gaussian", "poisson", "binomial", or "ordinal".
ordinal_mode	For ordinal models: "surprise" (default, based on observation probability) or "expected_score".
top_n_labels	Number of points to label on the plot. The ranking metric depends on the model type (see Description): deviation from the mean prediction for Gaussian/Poisson and the ordinal expected-score mode, absolute deviance residual for binomial, and surprise for the ordinal surprise mode. Default is 5.
strata_info	Optional data frame of strata labels, generally extracted from `maihda_model` objects.

Details

The vertical bars are 95% intervals for the plotted values. On a stratum panel each bar is an interval for the stratum's weighted mean prediction, the point drawn: posterior quantiles of that mean for a **brms** fit, and for an **lme4** fit a delta-method interval from the joint covariance of the fixed and random effects (Henderson's mixed-model equations), formed on the logit scale for a probability and the log scale for a count, so that a bar stays inside the range of the response. A case-level panel of an **lme4** fit draws the same interval for each row. The **lme4** interval treats the variance parameters as known: it runs narrow when strata are few or small or an outcome is rare, and a random effect whose variance is estimated at zero adds no uncertainty to it. Other models draw case-level bars from the standard errors of their `predict()` method and no stratum bars; **WeMix** and ordinal fits draw none.

Value

A **patchwork** object containing two **ggplot2** panels.

References

Henderson, C. R. (1975). Best linear unbiased estimation and prediction under a selection model. *Biometrics*, 31(2), 423-447. doi:10.2307/2529430

Bates, D., Maechler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1), 1-48. doi:10.18637/jss.v067.i01

Kackar, R. N., & Harville, D. A. (1984). Approximations for standard errors of estimators of fixed and random effects in mixed linear models. *Journal of the American Statistical Association*, 79(388), 853-862. doi:10.1080/01621459.1984.10477102

---

predict.maihda_model *Predict method for maihda_model objects*

Description

S3 `predict` method for MAIHDA models: a thin alias of `predict_maihda`, so `predict(model, type = "strata")` and `predict_maihda(model, type = "strata")` are interchangeable. See `predict_maihda` for the full documentation of the arguments and return value.

Usage

```
## S3 method for class 'maihda_model'
predict(
  object,
  newdata = NULL,
  type = c("individual", "strata", "response", "link"),
  scale = c("response", "link"),
  allow_new_levels = FALSE,
  ...
)
```

Arguments

`object`	A maihda_model object from `fit_maihda()`.
`newdata`	Optional data frame for making predictions. If NULL, uses the original data from model fitting.
`type`	Character string specifying prediction type: • "individual": Individual-level predictions including random effects • "strata": Stratum-level predictions (random effects only). For a longitudinal (growth-curve) fit a stratum is a *trajectory*, so this returns the per-stratum trajectory parameters (baseline deviation, random intercept and random slope(s)) rather than a single random effect. A *non-longitudinal* fit whose stratum random effects include random slopes (a hand-written `(1 + x | stratum)`) is an error here, as in `summary()`: the single value this would return is the intercept alone – the stratum effect only where the slope variables are zero.

"link" or "response" is also accepted and gives individual-level predictions on that scale.

`scale` Character string specifying the prediction scale for individual-level predictions: "response" (default) or "link". For a cumulative (ordinal) model the "link" scale is the latent location η and the "response" scale is the *expected category score* $\sum_k kP(Y = k)$ (categories scored 1..K in their declared order). For an aggregated-binomial fit (an lme4 `cbind(success, failure)` or a brms `success | trials(n)`) the "response" scale is the per-trial *probability* on both engines (not the expected success count).

`allow_new_levels`
Logical. `FALSE` (default): a stratum in `newdata` that the model never saw is an error, as in **lme4**. `TRUE`: an unseen stratum is predicted *at a zero random effect* for that stratum; any other random effect the row has (a `context` intercept, a longitudinal growth term) is kept. That is the prediction for a stratum whose effect is zero, **not** a response-scale population average: the two agree on the link scale and under an identity link, but under a log link with stratum variance τ^2 the population mean is larger by a factor $\exp(\tau^2/2)$, and under a logit link the population probability lies closer to 0.5. The response-scale value returned is the median, not the mean, of the stratum-specific means. Applies to `type = "individual"` only; with `type = "strata"` an unseen stratum is always an error.

`...` Additional arguments passed to predict method of underlying model.

Value

Depending on type:

- For "individual": A numeric vector of predicted values on the requested scale, one per row of `newdata` – or, when `newdata` is `NULL`, one per ANALYTIC row, matching `nrow(object$data)`. A model fitted with `na.action = na.exclude` is *not* padded back out to the original input rows the way `predict()` on the underlying fit is, so the result can always be bound to `object$data`.
- For "strata": A data frame with stratum ID and predicted random effect. When `newdata` is supplied, the result is restricted to the strata present in `newdata` (and a stratum the model never saw is an error, as for "individual"); when `newdata` is `NULL`, every training stratum is returned.

See Also

`predict_maihda`

`predict_maihda` *Predict from MAIHDA Model*

Description

Makes predictions from a fitted MAIHDA model, either at the stratum level or individual level.

Usage

```
predict_maihda(
  object,
  newdata = NULL,
  type = c("individual", "strata", "response", "link"),
  scale = c("response", "link"),
  allow_new_levels = FALSE,
  ...
)
```

Arguments

object	A maihda_model object from fit_maihda().
newdata	Optional data frame for making predictions. If NULL, uses the original data from model fitting.
type	Character string specifying prediction type: • "individual": Individual-level predictions including random effects • "strata": Stratum-level predictions (random effects only). For a longitudinal (growth-curve) fit a stratum is a *trajectory*, so this returns the per-stratum trajectory parameters (baseline deviation, random intercept and random slope(s)) rather than a single random effect. A *non-longitudinal* fit whose stratum random effects include random slopes (a hand-written (1 + x | stratum)) is an error here, as in summary(): the single value this would return is the intercept alone – the stratum effect only where the slope variables are zero. "link" or "response" is also accepted and gives individual-level predictions on that scale.
scale	Character string specifying the prediction scale for individual-level predictions: "response" (default) or "link". For a cumulative (ordinal) model the "link" scale is the latent location η and the "response" scale is the *expected category score* $\sum_k kP(Y = k)$ (categories scored 1..K in their declared order). For an aggregated-binomial fit (an lme4 cbind(success, failure) or a brms success | trials(n)) the "response" scale is the per-trial *probability* on both engines (not the expected success count).
allow_new_levels	Logical. FALSE (default): a stratum in newdata that the model never saw is an error, as in **lme4**. TRUE: an unseen stratum is predicted *at a zero random effect* for that stratum; any other random effect the row has (a context intercept, a longitudinal growth term) is kept. That is the prediction for a stratum whose effect is zero, **not** a response-scale population average: the two agree on the link scale and under an identity link, but under a log link with stratum variance τ^2 the population mean is larger by a factor $\exp(\tau^2/2)$, and under a logit link the population probability lies closer to 0.5. The response-scale value returned is the median, not the mean, of the stratum-specific means. Applies to type = "individual" only; with type = "strata" an unseen stratum is always an error.
...	Additional arguments passed to predict method of underlying model.

Value

Depending on type:

- For "individual": A numeric vector of predicted values on the requested scale, one per row of `newdata` – or, when `newdata` is `NULL`, one per ANALYTIC row, matching `nrow(object$data)`. A model fitted with `na.action = na.exclude` is *not* padded back out to the original input rows the way `predict()` on the underlying fit is, so the result can always be bound to `object$data`.
- For "strata": A data frame with stratum ID and predicted random effect. When `newdata` is supplied, the result is restricted to the strata present in `newdata` (and a stratum the model never saw is an error, as for "individual"); when `newdata` is `NULL`, every training stratum is returned.

Examples

```
strata_result <- make_strata(maihda_sim_data, vars = c("gender", "race"))
model <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata_result$data)

# Individual predictions
pred_ind <- predict_maihda(model, type = "individual")

# Stratum predictions
pred_strata <- predict_maihda(model, type = "strata")
```

`print.maihda_analysis` *Print a MAIHDA Analysis*

Description

Print a MAIHDA Analysis

Usage

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

Arguments

`x`	A `maihda_analysis` object from `maihda`.
`...`	Additional arguments (not used).

Value

No return value, called for side effects.

print.maihda_describe *Print a MAIHDA sample description*

Description

Print a MAIHDA sample description

Usage

```
## S3 method for class 'maihda_describe'
print(x, digits = x$digits, ...)
```

Arguments

x	A maihda_describe object from maihda_describe.
digits	Decimal places (defaults to the value stored on x).
...	Additional arguments (not used).

Value

Invisibly, x.

print.maihda_group_comparison *Print method for MAIHDA group comparisons*

Description

Print method for MAIHDA group comparisons

Usage

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

Arguments

x	A maihda_group_comparison object.
...	Additional arguments (not used).

Value

No return value, called for side effects.

print.maihda_ic *Print MAIHDA information criteria*

Description

Print MAIHDA information criteria

Usage

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

Arguments

x	A maihda_ic object from maihda_ic.
...	Additional arguments (not used).

Value

No return value, called for side effects.

print.maihda_interactions *Print a MAIHDA interaction diagnostic*

Description

Print a MAIHDA interaction diagnostic

Usage

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

Arguments

x	A maihda_interactions object from maihda_interactions.
...	Additional arguments (not used).

Value

No return value, called for side effects.

print.maihda_long_pcv *Print a longitudinal MAIHDA PCV*

Description

Print a longitudinal MAIHDA PCV

Usage

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

Arguments

x	A maihda_long_pcv object.
...	Unused.

Value

The object, invisibly.

print.maihda_model *Print method for maihda_model*

Description

Print method for maihda_model

Usage

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

Arguments

x	A maihda_model object
...	Additional arguments

Value

No return value, called for side effects.

print.maihda_pcv_importance *Print a PCV importance attribution*

Description

Print a PCV importance attribution

Usage

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

Arguments

x	A maihda_pcv_importance object from pcv_importance.
digits	Number of digits for the contribution columns. Default 4.
...	Additional arguments (not used).

Value

Invisibly, x.

print.maihda_stepwise *Print a stepwise MAIHDA table*

Description

Print a stepwise MAIHDA table

Usage

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

Arguments

x	A maihda_stepwise object from stepwise_pcv.
...	Additional arguments (not used).

Value

Invisibly, x.

print.maihda_strata *Print method for maihda_strata objects*

Description

Print method for maihda_strata objects

Usage

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

Arguments

x	A maihda_strata object
...	Additional arguments (not used)

Value

No return value, called for side effects.

print.maihda_summary *Print method for maihda_summary objects*

Description

Print method for maihda_summary objects

Usage

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

Arguments

x	A maihda_summary object
...	Additional arguments (not used)

Value

No return value, called for side effects.

print.maihda_table *Print a MAIHDA results table*

Description

Print a MAIHDA results table

Usage

```
## S3 method for class 'maihda_table'
print(x, digits = x$digits, ...)
```

Arguments

x	A maihda_table object from maihda_table.
digits	Decimal places (defaults to the value stored on x).
...	Additional arguments (not used).

Value

Invisibly, x.

print.pcv_result *Print method for PCV results*

Description

Print method for PCV results

Usage

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

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

Arguments

x	A pcv_result object from calculate_pcv
...	Additional arguments

Value

No return value, called for side effects.

run_maihda_app *Run MAIHDA Shiny Application*

Description

Launches a Shiny graphical user interface that exposes core functions of the MAIHDA package, allowing for visual data exploration, model fitting, and performance visualization.

Usage

```
run_maihda_app()
```

Value

No return value, called to launch the shiny app.

Examples

```
## Not run:
run_maihda_app()

## End(Not run)
```

stepwise_pcv *Stepwise Proportional Change in Variance (PCV)*

Description

Estimates the proportional change in variance (PCV) sequentially by fitting intermediate (partially-adjusted) models, adding each predictor one-by-one. The step-specific PCV is the change in between-stratum variance contributed by a predictor *given the variables already in the model*. Because the steps are sequential it is order-dependent: it reflects each variable's marginal, model-dependent change, not an order-invariant "unique" contribution.

Usage

```
stepwise_pcv(
  data,
  outcome,
  vars,
  engine = "lme4",
  family = "gaussian",
  context = NULL,
  sampling_weights = NULL,
  estimation = c("fitted", "ML")
)
```

Arguments

data	Data frame with observations. Ensure make_strata was run first so the stratum variable exists.
outcome	Character string; the dependent variable.
vars	Character vector; predictors (strata groupings & covariates) to add sequentially to the model.
engine	Modeling engine ("lme4", "brms", "wemix", or "ordinal"). Default is "lme4"; switches to "wemix" automatically when sampling_weights is supplied, and to "ordinal" for an ordinal family or ordered-factor outcome.
family	Error distribution and link function. Default is "gaussian".
context	Optional character vector naming one or more higher-level *context* columns in data (e.g. "school", "region"), forwarded to every step's fit_maihda call so that each model carries a crossed contextual random intercept, (1 | context), alongside the intersectional stratum effect – the stepwise analogue of maihda(context =) (the literature's contextual cross-classified MAIHDA). The reported Step_PCV / Total_PCV are then the between-stratum PCV **net of** the context (the context intercept is held in the null model and every step, so the stratum variance is isolated from it), and an extra Context_Variance column reports the between-context variance held at each step. Context columns join the complete-case filter so every step shares one analytic sample. A context variable may not be a stratum dimension, "stratum" itself, or one of vars. Supported by the lme4 and brms engines only (wemix and ordinal fit no crossed random effects). See fit_maihda for the full contextual-model semantics. Default NULL (no context).
sampling_weights	Optional name of a sampling-weight column for design-weighted stepwise fits; see fit_maihda. The weight column joins the complete-case filter so every step uses the same analytic sample.
estimation	Variance-estimation basis for the between-stratum variances compared across steps, "fitted" (default) or "ML"; see calculate_pcv. Affects Gaussian lmer fits only.

Details

All models are fit on the complete cases for outcome, stratum, and all variables in vars so that each sequential variance comparison uses the same analytic sample.

For a binary (or ordinal) outcome the sequential between-stratum variances live on the latent scale, whose level-1 variance is fixed by the link: a step that adds an *individual-level* variable (one varying within strata) rescales the latent metric itself, so its Step_PCV mixes explained variance with rescaling and can be understated or negative on that account – see the "Latent-scale families and rescaling" note in calculate_pcv. Steps that add a stratum-constant dimension are largely unaffected.

For a binary outcome the table additionally tracks discriminatory accuracy (Merlo et al. 2016): AUC is each model's C-statistic and Step_AUC / Total_AUC are its *absolute* change (delta-AUC), in contrast to the *proportional* Step_PCV / Total_PCV. The MOR is reported for the logit link (NA otherwise) and is a monotone transform of the between-stratum variance already in Variance. For

a design-weighted fit (`sampling_weights`) the AUC is the design-weighted (population) C-statistic. Reuses `maihda_discriminatory_accuracy` on each step's fitted model, so no additional models are fit. Note that adding a *stratum-defining* dimension (one already encoded by the strata) typically leaves the AUC essentially unchanged: it re-partitions the between-stratum variance (so the PCV and MOR move) but not the per-stratum predicted ranking the rank-based AUC depends on. The AUC trajectory is therefore most informative for individual-level covariates that vary *within* strata.

Value

A data.frame (class `maihda_stepwise`) showing the sequential models, the between-stratum variance at each step, and both the step-specific and total PCV. For a **binary** (binomial/Bernoulli) outcome it also carries the discriminatory-accuracy trajectory: `AUC` (the C-statistic of each step's model – step 0 is the strata-only discriminatory accuracy), `Step_AUC` and `Total_AUC` (the *absolute* change in AUC, delta-AUC, versus the previous step and versus the null), and `MOR` (the Median Odds Ratio, logit link only). These columns are absent for non-binary outcomes. When `context` is supplied, a `Context_Variance` column reports the between-context variance held at each step (for a `brms` fit the posterior median of each context's variance, summed over the contexts; `Variance` is a posterior median too). For a binary outcome the discriminatory-accuracy trajectory is still reported alongside it: the `AUC` and `MOR` are the intersectional-scope (between-stratum) quantities – the concordance of the fixed effects plus the stratum interaction, *excluding* the context random effect – so they carry the same net-of-context scope as the `Step_PCV` / `Total_PCV` columns, exactly as `summary.maihda_model` reports the intersectional-scope AUC for a contextual fit.

The variance-estimation basis chosen by `estimation` is recorded as an `"estimation"` attribute on the returned table; the basis actually used is recorded as `"estimation_used"` (`"mixed"` when `estimation = "ML"` but a step's ML refit was skipped at the boundary, leaving it on REML). Both are shown by `print()`. An `"ml_variances"` attribute is `TRUE` when the compared variances are maximum-likelihood estimates (`glmer`, wemix or ordinal fits, or `lmer` fits refitted by maximum likelihood); `print()` notes it under the default `estimation`, as `calculate_pcv` does.

`Step_PCV` is `NA` for any step whose *preceding* model has a between-stratum variance at the singularity boundary: that variance is the step's denominator, and at the boundary it carries only optimizer noise, so the ratio is undefined rather than small. Those steps are listed in an `"undefined_step_pcv"` attribute and noted by `print()`. `Total_PCV` is unaffected, since it divides by the null model's variance.

References

Merlo, J., Wagner, P., Ghith, N., & Leckie, G. (2016). An original stepwise multilevel logistic regression analysis of discriminatory accuracy: the case of neighbourhoods and health. *PLOS ONE*, 11(4), e0153778.

Examples

```
strata_result <- make_strata(maihda_sim_data, c("gender", "race"))
stepwise_pcv(strata_result$data, "health_outcome", c("gender", "race", "age"))

# Contextual cross-classified stepwise PCV: strata crossed with a higher-level
# context (country). Step_PCV / Total_PCV are then net of the country intercept,
# and a Context_Variance column reports the between-country variance per step.
cc <- make_strata(maihda_country_data, c("gender", "ses"))
```

```
stepwise_pcv(cc$data, "math", c("gender", "ses"), context = "country")
```

`summary.maihda_analysis` *Summarize a MAIHDA Analysis*

Description

Returns the variance summary (VPC/ICC, variance components, stratum estimates) of one of the analysis's models. The per-group comparison, when present, is attached as the `"groups"` attribute.

Usage

```
## S3 method for class 'maihda_analysis'
summary(object, which = c("null", "adjusted"), ...)
```

Arguments

`object`	A `maihda_analysis` object from `maihda`.
`which`	Which model to summarize: `"null"` (default) or `"adjusted"`. A crossed-dimensions analysis has a single model and accepts only `"null"`.
`...`	Additional arguments (not used). The summaries are computed by `maihda`, so arguments that would change them are rejected rather than ignored: set `bootstrap`, `n_boot`, `conf_level`, `response_vpc` and `seed` on `maihda` itself, and for a different fixed-effect reference summarise the fitted model directly, as in `summary(x$model_adjusted, df_method = "bootstrap")`.

Value

The `maihda_summary` for the requested model.

Which model

A two-model `maihda` analysis holds a **null** model (strata as the only predictor) and an **adjusted** model (the dimensions' additive main effects added as fixed effects), and this returns the null model's summary by default – the one the headline VPC/ICC comes from. The strata dimensions are the random-effect grouping there, not fixed effects, so the printed fixed-effect table shows only the intercept and any covariates. Use `which = "adjusted"` for the model whose fixed effects carry the dimensions' additive main effects (the same object as `object$summary_adjusted`), matching `tidy`'s `which` argument.

`summary.maihda_model` *Summarize MAIHDA Model*

Description

Provides a summary of a MAIHDA model including variance partition coefficients (VPC/ICC) and stratum-specific estimates.

Usage

```
## S3 method for class 'maihda_model'
summary(
  object,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  response_vpc = FALSE,
  seed = NULL,
  df_method = c("between-within", "normal", "bootstrap"),
  ...
)
```

Arguments

`object`	A maihda_model object from `fit_maihda()`.
`bootstrap`	Logical indicating whether to compute parametric bootstrap confidence intervals for VPC/ICC. Default is FALSE. Supported for lme4 models only; `brms` models always return a posterior credible interval (see Details), so `bootstrap = TRUE` is rejected for them. Each simulated response is refitted from the fitted model's estimates (`lme4::refit()`); for a growth fit (`fit_maihda(id =, time =)`) it is then refitted as `fit_maihda()` fits the model, from the further starting values and, for a `glmer` fit, across a variance bound (see `fit_maihda`, arguments `time` and `stratum_slope`), and the best fit is kept, at two to five times the time of one refit. For a negative-binomial model (`glmer.nb`) the bootstrap refits hold the dispersion parameter theta fixed at its original estimate, as `lme4::refit()` does, so the interval is conditional on the estimated theta (theta's own sampling uncertainty is not propagated). The `ordinal` (clmm) engine has no simulate/refit machinery, so `bootstrap = TRUE` is rejected there (use `engine = "brms"` for interval estimates). For a Gaussian model carrying lme4 precision `weights`, the simulated responses draw each residual at $\sigma/\sqrt{w_i}$, so the interval rests on the same σ^2/w_i semantics as the point estimate above.
`n_boot`	Number of bootstrap samples if bootstrap = TRUE. Default is 1000.
`conf_level`	Confidence level for the VPC/ICC interval – the lme4 bootstrap CI or the brms posterior credible interval. Default is 0.95.

response_vpc — Logical; for a binomial (lme4) model, also compute the response-scale VPC (maihda_vpc_response) and attach it as the vpc_response slot. It is estimated by simulation, so it is opt-in (default FALSE) and uses seed for reproducibility. For a Poisson or negative-binomial (lme4) model whose only random effect is the stratum intercept, the same slot receives the count-scale VPC of Leckie et al. (2020), a closed form that needs no seed. Ignored for other families, engines and random-effect structures.

seed — Optional integer seed for the response-scale VPC simulation when response_vpc = TRUE.

df_method — Reference distribution for the fixed-effect p-values and intervals of an lme4 fit: "between-within" (default) a t on containment degrees of freedom for a Gaussian fit and a z elsewhere, "normal" a z, "bootstrap" a null-restricted parametric bootstrap costing n_boot refits *per tested fixed-effect coefficient*. "bootstrap" is the reference to use for a GLMM term that is constant within a stratum, such as an adjusted model's dimension main effects. Every other engine uses a z regardless.

... — Additional arguments (not currently used).

Value

A maihda_summary object containing:

vpc — Variance Partition Coefficient (ICC); for lme4 with bootstrap = TRUE and for all brms models this includes ci_lower/ci_upper/conf_level. For a contextual cross-classified fit this is the *between-stratum* share of all unexplained variance (net of the context)

variance_components — Data frame of variance components. For a contextual cross-classified fit (fit_maihda(context =)) each context appears as its own Context: <name> row

longitudinal — For a longitudinal (growth-curve) fit, the time-varying summary: vpc_t (the VPC over a reporting grid, with bootstrap or posterior bands), the per-level variances over that grid, the stratum and individual covariance blocks, and the two *trajectory VPCs* vpc_intercept and vpc_slope described below, each with an interval in vpc_intercept_ci / vpc_slope_ci (a posterior credible interval for brms, a bootstrap interval for a bootstrapped lme4 fit, NA otherwise) and the basis in trajectory_vpc_method. NULL for a cross-sectional fit

context — For a contextual cross-classified fit, the stratum vs. context partition: per-context variances and shares, the contexts' total share (vpc_context_total, with an interval when bootstrapped or for brms), and the between-stratum share (vpc_stratum); for brms also each context's number of levels in the fitted data (context_levels), and print() names a context with fewer than 10, whose variance depends on the prior on its standard deviation, and the realised partition in realised: the same quantities for the effects of the strata and context levels in the data (see the section Realised strata and contexts of maihda); NULL otherwise

discriminatory_accuracy — For a binomial/Bernoulli outcome, the maihda_da object (AUC + MOR) from maihda_discriminatory_accuracy; NULL otherwise. A contextual fit (fit_maihda(context

=)) is included – its headline AUC is the intersectional-scope concordance that excludes the context random effect. NULL for a crossed-dimensions fit (whose headline here is the additive/interaction decomposition) and a longitudinal fit

count_vpc For a log-link count model, the approximation the level-1 variance came from, the marginal count lambda (and theta / lambda_effective for the negative binomial) it was evaluated at, the alternatives all three approximations give at that lambda (level1_variance is the one used), and low_count – TRUE below the $\lambda = 2$ threshold above which Nakagawa et al. (2017) report the three agree. These are plug-in values at a single lambda: on the likelihood engines that is exactly the number in variance_components, but a brms summary works draw by draw and reports the posterior median of the level-1 variance there, which differs slightly. NULL for every other family.

vpc_response The response-scale VPC (maihda_vpc_response) when response_vpc = TRUE for a binomial lme4 model, including a contextual fit (the context variance enters the VPC denominator), and the count-scale VPC for a Poisson or negative-binomial lme4 model whose only random effect is the stratum intercept; NULL otherwise (including for crossed-dimensions and longitudinal fits)

stratum_estimates
Data frame of stratum-specific random effects with labels if available

fixed_effects Fixed-effect estimates. For the lme4, WeMix and ordinal engines a data frame with term, estimate, se, statistic, df, p_value and the Wald interval lower/upper at conf_level; df is NA wherever the reference is a z. The WeMix standard errors are its sandwich (robust) ones. For brms, the brms::fixef() matrix (posterior mean, Est.Error and the credible-interval quantiles at conf_level). Available in a tidy, engine-independent shape from tidy(x, component = "fixed")

conf_level The interval level used for the fixed effects (and, when bootstrapped or Bayesian, the VPC)

df_method The reference distribution the fixed_effects table used, "between-within", "normal" or "bootstrap"

thresholds For a cumulative (ordinal) clmm fit, the threshold (cut point) estimates with standard errors – the cumulative model's "intercepts"; NULL otherwise

model_summary Original model summary

diagnostics Fit-quality diagnostics (singular fit / convergence) carried over from the fitted model and reported by the print method

strata_autobin_info
The auto-binning recipe carried over from the fitted model: for each numeric stratum dimension make_strata() discretised, its breaks and labels. The cut-points are quantiles of the analytic sample, so they define the strata (and hence the estimand); the print method reports them. An empty list when nothing was binned

Interpreting the VPC/ICC

The VPC is the between-stratum variance divided by the total *unexplained* variance. For the canonical single-stratum model that denominator is between-stratum + residual, but if the model includes additional random effects (e.g. (1 | site)) their variance is included in the denominator

too (between-stratum + other random effects + residual), so the VPC is the between-stratum *share* of all unexplained variance. It is a conditional/residual ICC that excludes variance captured by the fixed effects, so for models with covariates it is conditional on them. It is most commonly read from the null model `outcome ~ 1 + (1 | stratum)`, where it is the total between-stratum share. For non-Gaussian families the level-1 (residual) variance uses a latent/distributional approximation ($\pi^2/3$ for logistic; $\log(1+1/\lambda)$ for Poisson per Stryhn et al. 2006 and $\log(1+1/\lambda+1/\theta)$ for the negative binomial per Nakagawa, Johnson & Schielzeth 2017 – their `"delta"` and `"trigamma"` alternatives are available via `fit_maihda(count_approximation = )`, and the choice is reported in the printed summary and in `count_vpc` because the three diverge materially below a marginal count of 2 – each evaluated at a single *marginal* expected count λ: the mean over the analytic sample of the row-level $\lambda_i = \exp(x_i'\beta + v_i/2)$ – the fixed-part prediction with the log-normal correction for the row's total random-effect variance v_i. The counts are averaged *before* the transform, which is where the cited λ is defined and which reduces to Nakagawa et al.'s $\lambda = \exp(\beta_0 + \sigma^2/2)$ in the null model; *not* at the conditional fitted means, whose BLUPs would tie the level-1 variance to the realized random effects), so the VPC is on that latent scale; for a *weighted* Gaussian model the level-1 variance is the mean conditional residual variance, $\sigma^2/\bar{w}_i$, since the per-observation residual variance is σ^2/w_i. The stratum random effects represent the total between-stratum deviation; they equal the *pure* intersectional (interaction) component only when the additive main effects of the strata variables are included in the model.

For a Poisson or negative-binomial model the VPC can also be computed on the count scale itself, with no approximation of the level-1 variance (Leckie et al., 2020): see `maihda_vpc_response`, or set `response_vpc = TRUE`.

Fixed-effect reference distribution

A Gaussian `lme4` fit refers each Wald statistic to a t on *containment* (between-within) degrees of freedom, reported in the `df` column: a term absorbed by a random-effect grouping is tested against that grouping's units minus the terms it absorbs, and a term absorbed by none against n minus the random-effect levels. A random slope counts as absorbing. Set `df_method = "normal"` for a z instead.

A GLMM, a WeMix pseudo-ML fit and an `ordinal::clmm` fit have no finite-sample t and use the Wald z; a brms summary reports the posterior. For Kenward-Roger or Satterthwaite, apply **pbkrtest** or **lmerTest** to `x$model`.

`df_method = "bootstrap"` replaces that reference for an `lme4` fit with at least one fixed-effect term, and is the one to use for a GLMM, whose z is anticonservative for a term constant within a stratum, most of all when the strata are few. For each fixed-effect coefficient the model is refitted with that coefficient *constrained to zero*, `n_boot` responses are simulated from the restricted fit, the full model is refitted on each, and the observed Wald statistic is referred to the resulting distribution of $|t^*|$. The restricted fit of a growth model is made as `fit_maihda()` makes the model's own fit, with the same refits from further starting values (`fit_maihda`, arguments `time` and `stratum_slope`), and so is the full model's refit on each draw (see `bootstrap`). The estimate and standard error are unchanged. The p-value is $(1 + \#\{|t^*| \geq |t|\})/(B+1)$ over the B refits that succeeded, and the interval is the estimate plus or minus a critical value read from the same draws, times the standard error, so zero falls outside it exactly when the p-value is at most `1 - conf_level`. That agreement is algebraic, not a coverage guarantee. `df` is `NA`, and so are the intercept's p-value and interval: an intercept has no null model to simulate from.

The restriction is on the coefficient, not on its term. For a term spanning several design columns (a factor with three or more levels, a polynomial, an interaction between factors) each column is tested

against its own null: that column zeroed, the siblings kept and re-estimated. What the constraint means follows the fitted contrasts: treatment coding merges that level into the reference, sum coding sets its deviation from the unweighted mean of the level means to zero.

The bootstrap is an approximation, not an exact test. Its null is a fit estimated from the same data, so it is least reliable with few strata, where the stratum variance is often estimated at exactly zero; a p-value near the threshold from such a fit deserves little weight. `n_boot` sets the Monte Carlo resolution, not that approximation: the p-value lies on a grid of step $1/(B+1)$, and more draws only shrink its Monte Carlo error. No `n_boot` makes the test exact.

It costs `n_boot` refits *per tested coefficient* – a k-level factor costs $k-1$ blocks – and is separate from the `bootstrap = TRUE` VPC interval. A Gaussian refit takes milliseconds, a binomial one about a second at $n = 1000$ and tens of seconds at $n = 6000$, so `n_boot = 199` is the usual choice for a GLMM. `summary()` warns when the draws are too few for the requested level; the check is level-aware: 99, 199 and 999 are the smallest counts that put ten draws beyond the cut-off at the 10%, 5% and 1% levels.

Two VPCs for a longitudinal fit

A longitudinal summary reports **two different variance partitions**, and they are not interchangeable. They differ in one term of the denominator: the level-1 (occasion) residual variance σ^2_e, which in a growth model is *within-individual volatility* – how far a single measurement falls from that person's own smooth trajectory. It mixes measurement error, genuine short-term fluctuation, and any misfit of the assumed functional form. A cross-sectional MAIHDA cannot separate it from between-individual variance at all; repeated measures are what split the two.

The headline VPC (`vpc`, and `longitudinal$vpc_t` over time) keeps it:

$$VPC_S(t) = \frac{Var_S(t)}{Var_S(t) + Var_I(t) + \sigma^2_e}.$$

This is the discriminatory-accuracy question – how much of an *observed measurement* at time t a stratum accounts for – and it is the quantity comparable to published cross-sectional MAIHDA VPCs. Report this one unless you specifically mean the trajectory question.

The trajectory VPCs (`longitudinal$vpc_intercept` and `longitudinal$vpc_slope`) drop it, following Bell et al. (2024), equation (5):

$$VPC_{intercept} = \frac{Var_S(t_0)}{Var_S(t_0) + Var_I(t_0)}, \qquad VPC_{slope} = \frac{SlopeVar_S(t_0)}{SlopeVar_S(t_0) + SlopeVar_I(t_0)}.$$

These ask how intersectionally patterned people's *trajectories* are i.e., what share of the between-individual variation in where a trajectory starts, and in how fast it changes, lies between strata. Because σ^2_e is absent neither is affected by how noisy the outcome measure is, which makes them comparable across studies using different instruments.

Both are evaluated at the baseline t_0 (`ref_time`, the earliest observed time), pairing with `PCV_intercept` and `PCV_slope` from `maihda(decomposition = "longitudinal")`. The intercept VPC depends on where time is zeroed and the slope VPC does not, so an intercept VPC computed at another reference time, such as the mean age of Bell et al.'s examples, differs from the one reported here. `vpc_slope` is `NA` when the model was fit with `stratum_slope = FALSE` (no between-stratum slope variance exists to take a share of).

Both come with an interval in `vpc_intercept_ci` / `vpc_slope_ci`, and `trajectory_vpc_method` records its basis. For a `brms` fit the two shares are computed *per posterior draw* and reported

as the posterior median with a credible interval, matching `vpc_t` and the headline VPC on the same fit; for an `lme4` fit the point estimates are the plug-in from the fitted covariance blocks and `summary(bootstrap = TRUE)` adds a parametric-bootstrap interval (`NA` without one). Report the interval: these shares are poorly determined when the strata are few.

Note

For `lme4` models a VPC/ICC interval is obtained from a parametric bootstrap (`bootstrap = TRUE`). For `brms` models the VPC/ICC is summarised directly from the posterior draws: the reported estimate is the posterior median of the per-draw VPC ($E[\sigma^2]$-based, not the biased $E[\sigma]^2$) and the interval is a central credible interval at `conf_level` (default 95%), so no `bootstrap` argument is needed. The variance-components table of a cross-sectional `brms` fit reports the posterior median of each variance component – the between-stratum row is the variance the PCV and the Median Odds Ratio use, and its `sd` the posterior median SD – with the total and the proportions computed from those rows. The stratum proportion therefore equals the headline VPC when the level-1 variance is fixed by the link and the stratum is the only random effect, and differs from it slightly otherwise, because the median of a ratio is not the ratio of medians; in a crossed-dimensions fit, whose between-strata variance is the sum of several components, the share those rows add up to can differ markedly from the headline VPC when dimensions have few levels. With few strata the posterior of a variance is right-skewed, and its median overstates the variance (and with it the VPC and the Median Odds Ratio); the overstatement shrinks as strata are added. A longitudinal fit's table holds the posterior means of its covariance blocks. For non-Gaussian `brms` families the level-1 (residual) variance uses the usual latent-scale approximation. For the count families (Poisson, negative binomial) the marginal expected counts are propagated *per draw* – from the fixed-part linear-predictor draws and each draw's total random-intercept variance – for the intercept-only VPC structures (strata, crossed-dimensions, contextual), so the credible interval reflects fixed-effect and random-effect variance uncertainty; the negative-binomial `shape` draws are always propagated. A random-slope (longitudinal) structure, whose per-row random-effect design that fast path does not carry, instead holds the marginal expected counts at a posterior-mean plug-in (constant across draws) to avoid an expensive $ndraws \times nobs$ computation.

References

Bell, A., Evans, C., Holman, D., & Leckie, G. (2024). Extending intersectional multilevel analysis of individual heterogeneity and discriminatory accuracy (MAIHDA) to study individual longitudinal trajectories, with application to mental health in the UK. *Social Science & Medicine*, 351, 116955. doi:10.1016/j.socscimed.2024.116955

Leckie, G., Browne, W. J., Goldstein, H., Merlo, J., & Austin, P. C. (2020). Partitioning variation in multilevel models for count data. *Psychological Methods*, 25(6), 787-801. doi:10.1037/met0000265

Examples

```
strata_result <- make_strata(maihda_sim_data, vars = c("gender", "race"))
model <- fit_maihda(health_outcome ~ age + (1 | stratum), data = strata_result$data)
summary_result <- summary(model)

# With bootstrap CI
# summary_boot <- summary(model, bootstrap = TRUE, n_boot = 50)
```

`theme_maihda` *MAIHDA plot theme*

Description

A **ggplot2** theme that applies the MAIHDA brand identity to plot *chrome* only – navy titles and axis labels and soft slate gridlines. It encodes no data: the colourblind-safe data palettes used by the package's plotting functions are left untouched.

Usage

```
theme_maihda(base_size = 11, base_family = NULL)
```

Arguments

`base_size`	Base font size in points. Default 11.
`base_family`	Base font family. Defaults to `getOption("maihda.font", "")`, i.e. the graphics device default unless you have opted into a brand font globally. Pass a string to force a family for a single plot.

Details

Built on `ggplot2::theme_minimal()`, so it composes with additional `+ theme()` calls in the usual way (later settings win).

Value

A **ggplot2** theme object that can be added to a plot with +.

Fonts

By default the theme uses the graphics device's default font, which is safe on every device (including the PostScript/PDF devices used by `R CMD check` and many rendering back-ends). To render figures in the brand font (Montserrat) – matching the hex logo – set `options(maihda.font = "Montserrat")` or pass `base_family = "Montserrat"`, and use a graphics device that can resolve that family (e.g. **ragg** or **showtext**). Forcing a font that the active device cannot find produces "invalid font type" errors, which is why it is opt-in rather than automatic.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(mpg, wt)) +
  geom_point(colour = "#0072B2") +
  labs(title = "MAIHDA brand theme", x = "MPG", y = "Weight") +
  theme_maihda()
```

Index

∗ **datasets**
maihda_country_data, 32
maihda_health_data, 42
maihda_long_data, 50
maihda_sim_data, 54
maihda_sparse_data, 55

calculate_pcv, 3, *11, 12*, *26–28*, *45*, *49*, *58*, *67–69*, *89*, *91, 92*
compare_maihda, 7, *13*, *45*, *71*
compare_maihda_groups, 9, *24*, *26, 27*, *30*, *32*, *73*

fit_maihda, *10, 11*, 13, *24–28*, *30*, *33–36*, *38, 39*, *42, 43*, *45*, *49*, *52, 53*, *56, 57*, *59–61*, *64*, *66*, *91*, *94*, *97*

ggplot2::theme_minimal(), *100*
ggsave, *60*
glance, *41*
glance.maihda_analysis, *60*
glance.maihda_analysis *(*maihda_glance*)*, 41
glance.maihda_model *(*maihda_glance*)*, 41
glance.maihda_summary *(*maihda_glance*)*, 41
glm, *39, 40*

labs, *77*

maihda, *7*, *11*, *17*, *20*, 23, *32–35*, *37*, *42, 43*, *45, 46*, *49*, *55*, *57–60*, *64*, *69, 70*, *75*, *83*, *93*, *95*
maihda_adjusted_terms, *11*
maihda_auc, 31, *41*
maihda_country_data, 32
maihda_cumulative, *16*, 34, *53*
maihda_describe, 35, *72*, *84*
maihda_discriminatory_accuracy, *36*, 38, *52*, *58*, *63*, *92*, *95*
maihda_glance, 41
maihda_health_data, 42
maihda_ic, *4*, *8*, 43, *85*
maihda_interactions, *19*, *27, 28*, 45, *70*, *75, 76*, *85*
maihda_long_data, 50
maihda_mor, *41*, 51
maihda_proportional_odds_test, *22*, 52
maihda_sim_data, 54
maihda_sparse_data, 55
maihda_table, *35*, *38*, 56, *89*
maihda_tidiers, *42*, 58
maihda_upset_size, 60
maihda_vpc_response, *20*, *26*, 61, *95*, *97*
make_strata, *10*, *14*, *16*, *24*, *26*, *28, 29*, *35, 36*, *38*, *60*, 63, *66*, *91*

p.adjust, *20*, *27*, *46*
pcv_importance, 65, *78*, *87*
plot, *38*, *70, 71*
plot.maihda_analysis, 69
plot.maihda_comparison, 71
plot.maihda_describe, 72
plot.maihda_group_comparison, *13*, 73
plot.maihda_model, *60*, 74
plot.maihda_pcv_importance, 77
plot_prediction_deviation_panels, 78
predict, *80*
predict.maihda_model, 80
predict_maihda, *80*, 81, *81*
print.maihda_analysis, 83
print.maihda_describe, 84
print.maihda_group_comparison, 84
print.maihda_ic, 85
print.maihda_interactions, 85
print.maihda_long_pcv, 86
print.maihda_model, 86
print.maihda_pcv_importance, 87
print.maihda_stepwise, 87
print.maihda_strata, 88
print.maihda_summary, 88

print.maihda_table, 89
print.pcv_result, 89
print.pvc_result *(*print.pcv_result*)*, 89

refitML, *4*, *6*, *43*
run_maihda_app, 90

stepwise_pcv, *5*, *7*, *65–67*, *69*, *87*, 90
summary, *59*
summary.maihda_analysis, 93
summary.maihda_model, *4–6*, *16–18*, *20*, *26*, *27*, *30*, *49*, *52*, *57–59*, *61*, *63*, *92*, 94

theme_maihda, 100
tidy, *58*, *93*
tidy.maihda_analysis *(*maihda_tidiers*)*, 58
tidy.maihda_model *(*maihda_tidiers*)*, 58
tidy.maihda_summary *(*maihda_tidiers*)*, 58