

Package: magmaanlab (via r-universe)

October 27, 2026

Title Methods-development R bindings for the magmaan C++ SEM library

Version 0.2.2

License MIT + file LICENSE

Encoding UTF-8

LinkingTo Rcpp, RcppEigen, nloptr

Depends R (>= 4.3.0)

Imports Rcpp, nloptr, stats, utils

Suggests testthat (>= 3.0.0), lavaan

Config/testthat/edition 3

SystemRequirements C++23, USE_C17, NLOpt (a system NLOpt; on Unix, the library in nloptr is used automatically when no system NLOpt is found)

Description Thin Rcpp wrappers exposing magmaan's C++ functions for methods development and research, including frontier methods. The opinionated one-call package for ordinary users is 'magmaan', which builds on this one. Self-contained: the C++ core (plus the vendored PORT and QUADPACK routines) is compiled from sources vendored under cpp/src/ and linked against a system NLOpt, so the package installs without CMake or a prebuilt library. Not CRAN-bound.

Config/roxygen2/version 8.0.0

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake

Repository <https://jonasmoss.r-universe.dev>

Date/Publication 2026-10-27 15:59:30 UTC

RemoteUrl <https://github.com/JonasMoss/magmaan>

RemoteRef v0.2.2

RemoteSha c0dc6e296b515a3aff24353c390fd53e964de027

RemoteSubdir r-package

Contents

association_ml_global_test	4
association_ml_ij	5
association_ml_modification_indices	6
bounds_loading	7
bounds_pos_var	7
bounds_standard	8
bounds_variance	8
bounds_wide	9
case_rerun	9
composite_weights	10
compute_defined	10
continuous_invariance	11
convention_fit_measures	12
convention_inference	12
convergence_sim	13
convergence_sim_catalog	14
convergence_sim_dejonckere_crossloading	14
convergence_sim_dejonckere_msst	15
convergence_sim_dejonckere_shrinkage	15
convergence_sim_dejonckere_simple	16
convergence_sim_ludtke_cfa	17
df_to_data	17
df_to_fcsem_data	18
df_to_fiml_data	19
eqs_model	19
est_change	20
est_change_approx	21
est_change_raw	22
est_change_raw_approx	23
factor_score_precision	24
factor_scores	24
fcsem_fit_measures	25
fcsem_model_spec	25
fcsem_standard_errors	26
fcsem_standardized_rows	26
fit_index_intervals_misspec	27
fit_measures	28
fit_measures_change	28
fit_measures_misspec	29
fit_measures_misspec_mixed_ordinal	30
fit_measures_noniterative	31
fit_ml_fcsem	32
fit_model	32
fit_noniterative_cfa	34
fit_noniterative_cfa_metric	36
fit_noniterative_cfa_restricted	37
fit_twolevel	39

fmg_nested	40
fmg_nested_ordinal	41
fmg_pvalues	42
fmg_tests	43
fmg_tests_ordinal	44
frontier_fit_fiml_multiinfo	45
frontier_fit_fiml_psd	46
frontier_fit_gls_psd	46
frontier_fit_mixed_ordinal_psd	47
frontier_fit_ml_multiinfo	48
frontier_fit_ml_psd	49
frontier_fit_ml_psd_fallback	50
frontier_fit_ml2s_psd	51
frontier_fit_ordinal_psd	52
frontier_fit_sphere	53
frontier_fit_uls_psd	54
frontier_fit_wls_psd	55
frontier_newton_accuracy	56
frontier_reidentify	57
global_score_flip_test	57
guttman_h	59
inference_rows	60
lab_inference_primitives	60
lav_residuals	62
magmaan_core	63
magmaan_fcsem	63
mahalanobis_rerun	64
mi_screen	64
model_spec	65
modification_indices	66
modification_indices_lrt	67
modification_indices_robust	67
mplus_data	70
mplus_model	71
mplus_wlsmv_invariance	73
nested_score_test	74
noniterative_cfa_constrained	75
noniterative_cfa_difference_test	76
noniterative_cfa_grouped_inference	77
noniterative_cfa_inference	78
noniterative_cfa_modification_indices	79
noniterative_cfa_pseudo_lrt	80
noniterative_cfa_scalar_invariance	81
noniterative_cfa_se	82
noniterative_cfa_wald	82
parameter_table	83
policy_fit_measures	84
policy_inference	85

policy_mi_refit	85
policy_modification_indices	86
policy_nested	87
prepare_inference_data	87
prepare_model	89
refit_from_null	92
robust_nested_lrt	93
robust_ordinal_ij	95
sam	96
score_flip_test	97
score_tests	98
score_tests_lrt	99
scores	100
sim_model_batch	102
sim_model_calibrate	103
sim_model_draw	103
sim_ordcorr_batch	104
sim_ordcorr_calibrate	105
sim_ordcorr_draw	106
sim_ordcorr_mg_batch	107
sim_ordcorr_mg_calibrate	108
sim_ordcorr_mg_draw	109
sim_ordcorr_mg_summary_calibrate	109
sim_ordcorr_summary_calibrate	110
standardized	111
standardized_rows	111
structural_identification	112
vcov.magmaan_fit	112
version	114

Index 115

association_ml_global_test

Lab association ML global and nested spectral tests

Description

Composes exact empirical first-stage influence with the local correlation ML residual metric for global tests, or observed sensitivity and exact parameter restrictions for nested tests. Reports All, SB and PEBA4 references. Ordinary inference remains unavailable.

Usage

```
association_ml_global_test(fit, ordinal_stats)
association_ml_nested_test(fit_alt, fit_null, ordinal_stats)
```

Arguments

`fit`, `fit_alt`, `fit_null` Association ML fitted models; alternative first.

`ordinal_stats` Common complete all-ordinal Stage-1 statistics with raw rows.

Value

A list with availability and a typed reason, statistic, degrees of freedom, spectrum, three references and reconstruction matrices. Zero degrees of freedom has reason **zero_df** and undefined p-values. Moment nesting, penalties, boundaries and unsupported Stage-1 channels are refused.

<code>association_ml_ij</code>	<i>Exact sampling sandwich for ordinal association ML</i>
--------------------------------	---

Description

Lab-only evaluation-point inference for the Gaussian discrepancy on estimated polychoric correlations. Uses empirical-Jacobian first-stage rows, observed sensitivity including correlation-map curvature, and fixed-allocation group sampling. Ordinary inference remains unavailable.

Usage

```
association_ml_ij(fit, ordinal_stats)
```

Arguments

`fit` An all-ordinal association-ML fit in DELTA coordinates.

`ordinal_stats` The complete-row Stage-1 statistics used for fitting.

Details

Requires complete all-ordinal rows, saturated thresholds, fixed response scales, linear association constraints, an identified model, positive observed sensitivity and interior primitive covariance matrices. Missing rows, mixed indicators, nonlinear constraints, covariance boundaries and penalties are unsupported. Sensitivity is computed by central differences of the analytic score; no global, nested or modification-index reference law is supplied.

Value

A list with **available**; unavailable results include **reason** and optionally **detail**. Available results contain the objective **value**, active **score**, observed **H**, meat **B**, coordinate map **K**, group target derivatives **D**, **vcov_active**, full **vcov**, and group lists **influence_active** and **influence**. Influence rows are unscaled: summing their cross-products and dividing by total sample size squared reconstructs the covariance. Full rows include saturated thresholds and their cross-covariance with association parameters.

association_ml_modification_indices
Lab association-ML modification indices and linear releases

Description

One-dimensional score tests using observed sensitivity for nuisance projection and exact empirical Stage-1 influence for the score variance.

Usage

```
association_ml_modification_indices(fit, ordinal_stats, absent = TRUE)
association_ml_score_tests(fit, ordinal_stats)
```

Arguments

fit	An all-ordinal association-ML lab fit with complete raw rows.
ordinal_stats	The same Stage-1 statistics and integer rows used to fit.
absent	Enumerate absent loadings, covariances and structural regressions among equation variables in addition to fixed rows.

Details

The robust statistic is N times the squared efficient score divided by its empirical meat, referenced to chi-square with one degree of freedom. EPC uses the observed Schur sensitivity. The unscaled **mi** uses that same sensitivity. Equality rows index the original affine constraint system; EPC is the coefficient along its normalized release direction, with **epc_full** giving the full parameter displacement. Fixed-row EPC is in the candidate parameter's units. No joint multi-direction release is exposed. Threshold, mean and response-scale releases, singular information, covariance boundaries, penalties, missing data and mixed indicators remain unavailable. Ordinary association inference remains guarded pending policy calibration.

Value

A list with **available**, **metric = "observed"** and **rows**. Each row retains availability or a typed reason. Available rows contain **statistic**, **df**, **p_value**, **epc**, **epc_full**, **mi** and diagnostic score, sensitivity, meat and coordinate matrices.

bounds_loading	<i>Parameter bounds: bounds_loading</i>
----------------	---

Description

Constructs parameter bounds in model free-parameter order. Sample-based bounds use the supplied data summaries.

Usage

```
bounds_loading(model, data)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Value

A list of lower and upper parameter-bound vectors.

bounds_pos_var	<i>Parameter bounds: bounds_pos_var</i>
----------------	---

Description

Constructs parameter bounds in model free-parameter order. Sample-based bounds use the supplied data summaries.

Usage

```
bounds_pos_var(model)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
--------------	--

Value

A list of lower and upper parameter-bound vectors.

bounds_standard	<i>Parameter bounds: bounds_standard</i>
-----------------	--

Description

Constructs parameter bounds in model free-parameter order. Sample-based bounds use the supplied data summaries.

Usage

```
bounds_standard(model, data)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Value

A list of lower and upper parameter-bound vectors.

bounds_variance	<i>Parameter bounds: bounds_variance</i>
-----------------	--

Description

Constructs parameter bounds in model free-parameter order. Sample-based bounds use the supplied data summaries.

Usage

```
bounds_variance(model)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
--------------	--

Value

A list of lower and upper parameter-bound vectors.

bounds_wide	<i>Parameter bounds: bounds_wide</i>
-------------	--------------------------------------

Description

Constructs parameter bounds in model free-parameter order. Sample-based bounds use the supplied data summaries.

Usage

```
bounds_wide(model, data)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Value

A list of lower and upper parameter-bound vectors.

case_rerun	<i>Refit a model repeatedly, each time with one case removed</i>
------------	--

Description

The exact leave-one-out engine: for each selected case, drop its row, rebuild the sample statistics, and refit (warm-started from the full-sample solution). Mirror of `semfindr::lavaan_rerun()`. Continuous ML/ULS/GLS.

Usage

```
case_rerun(fit, data = NULL, to_rerun = NULL, warm_start = TRUE)
```

Arguments

fit	A fitted <code>magmaan_fit</code> (continuous ML/ULS/GLS).
data	Optional original data frame. Required for multiple-group fits; for single-group fits, defaults to the raw data on <code>fit</code> .
to_rerun	Integer case indices to refit; default all cases.
warm_start	Seed each refit from the full-sample estimates (faster, and keeps refits on the same optimum). Default <code>TRUE</code> .

Details

Single-group fits work directly from the raw data the fit carries. For multiple-group fits, pass the original **data** frame (the per-group raw blocks on the fit have lost the original row order needed to map a global case index to its group); case ids are then the data-frame rows, as in `semfindr`.

Value

A `magmaan_case_rerun` list: `rerun` (per-case refit fits, NULL when a refit fails or is inadmissible), `fit`, `selected`, `case_id`, `converged`.

<code>composite_weights</code>	<i>Composite score weights</i>
--------------------------------	--------------------------------

Description

Computes composite weights and uncertainty from a fit and an explicitly supplied parameter covariance.

Usage

```
composite_weights(fit, vcov)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>vcov</code>	Explicit covariance matrix of free parameters, in theta order.

Value

Native composite-weight results.

<code>compute_defined</code>	<i>Evaluate defined model parameters</i>
------------------------------	--

Description

Evaluates `:=` expressions and their delta-method uncertainty using an explicit parameter covariance.

Usage

```
compute_defined(model, fit, vcov)
```

Arguments

<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>fit</code>	A fitted magmaan model.
<code>vcov</code>	Explicit covariance matrix of free parameters, in theta order.

Value

A table of defined parameters and uncertainty summaries.

`continuous_invariance`

Fit a continuous-data measurement-invariance ladder.

Description

The scalar rung explicitly frees every non-reference-group latent mean. This is required when observed intercepts are constrained equal: the scalar model is not an affine parameter subset of the metric model, so its robust difference test uses the delta restriction map.

Usage

```
continuous_invariance(
  model,
  data,
  group = NULL,
  steps = c("configural", "metric", "scalar"),
  estimator = c("ML", "FIML"),
  group_labels = NULL,
  optimizer = "nlopt-lbfgs",
  control = NULL,
  bounds = NULL
)
```

Arguments

<code>model</code>	A lavaan-style syntax string or a <code>magmaan_model_spec</code> carrying its source syntax.
<code>data</code>	A data frame containing the grouping column and continuous observed variables.
<code>group</code>	Name of the grouping column. When <code>model</code> is a grouped model spec, its grouping column is used by default.
<code>steps</code>	Requested rungs from "configural", "metric", and "scalar". Requesting "scalar" also fits its metric prerequisite.
<code>estimator</code>	"ML" for complete data or "FIML" when incomplete data require full-information estimation.

group_labels Optional group ordering. Defaults to data order (or the supplied model spec's ordering).

optimizer, control, bounds Fitting controls forwarded to `fit_model()`.

Value

A `magmaan_continuous_invariance` list with fitted models, robust nested tests, and the constructed specifications.

convention_fit_measures

Lavaan-compatible fit measures

Description

Composes standard, scaled and robust fit-index families in C++, including a matching independence baseline, noncentral chi-square intervals and close-fit p-values. Estimates are retained.

Usage

```
convention_fit_measures(fit, convention)
```

Arguments

fit A fitted magmaanlab model retaining observations or ordinal stats.

convention A named lavaan inference bundle.

Value

A data frame with index, estimate and reason columns and a `lavaan_compat` attribute. Incompatible bundles error. FIML ML/MLR use saturated H1 EM parameter tolerance 1e-10.

convention_inference *Named lavaan inference on retained estimates*

Description

Thin adapters to C++ compatibility composers. A convention changes the reported inference without changing fitting. Components outside the checked model/data slice return an unavailable reason. Complete mixed DWLS supports "WLSMV": expected-bread NA-COV covariance, scaled-shifted global and Satorra-2000 delta nested tests. Missing mixed observations and other mixed bundles are unsupported. Delta/theta reporting is point-gated; grouped/theta retained endpoints have limited validation due to optimizer stopping differences.

Usage

```
convention_inference(fit, convention)

convention_nested(fit_H1, fit_H0, convention)
```

Arguments

fit A fitted magmaan_fit.

convention A lavaan bundle: "ML", "MLM", "MLR", "DWLS", "WLSMV", "WLSM" (unsupported), "ULS", "ULSMV" or "WLS".

fit_H1, fit_H0 Alternative and restricted fits to the same data.

Value

A list with covariance and test results, component availability and convergence metadata. Nested inference returns one difference test.

convergence_sim	<i>Convergence simulation design: convergence_sim</i>
------------------------	---

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim(design = convergence_sim_catalog())$design, ...)
```

Arguments

design Design name from convergence_sim_catalog.

... Additional options forwarded to model construction or the underlying named operation.

Value

The design catalog, or generated data and model metadata for the selected design.

convergence_sim_catalog
<i>Convergence simulation design: convergence_sim_catalog</i>

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim_catalog()
```

Value

The design catalog, or generated data and model metadata for the selected design.

convergence_sim_dejonckere_crossloading
<i>Convergence simulation design: convergence_sim_dejonckere_crossloading</i>

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim_dejonckere_crossloading(n = 20L, variant = c("2025", "2022"), fit_model = c("correct", "omit_lambda8_lambda9", "omit_all_crossloadings"), beta = NULL, seed = NULL)
```

Arguments

n	Number of observations to generate.
variant	Published design version, 2025 or 2022.
fit_model	Crossloading specification to fit: correct or one of the omission variants shown in usage.
beta	Latent regression coefficient or coefficient vector; NULL uses the design default.
seed	Optional random-number seed.

Value

The design catalog, or generated data and model metadata for the selected design.

convergence_sim_dejonckere_msst

Convergence *simulation* *design:* *conver-*
gence_sim_dejonckere_msst

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim_dejonckere_msst(n = 10L, state_loadings = c(1, 0.5, 1.3), trait_loadings = c(1,
  measurement_error = c(0.6, 0.25, 0.7), trait_var = 0.6, state_residual_var = c(0.6,
    0.25, 0.7), seed = NULL)
```

Arguments

n Number of observations to generate.

state_loadings State-factor loading vector.

trait_loadings Trait-factor loading vector.

measurement_error Indicator measurement-error variance vector.

trait_var Trait-factor variance.

state_residual_var State-factor residual variance vector.

seed Optional random-number seed.

Value

The design catalog, or generated data and model metadata for the selected design.

convergence_sim_dejonckere_shrinkage

Convergence *simulation* *design:* *conver-*
gence_sim_dejonckere_shrinkage

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim_dejonckere_shrinkage(n = 20L, study = c("study1", "study2"), indicators = NULL, seed = NULL)
```

Arguments

n	Number of observations to generate.
study	Shrinkage design: study1 or study2.
indicators	Number of indicators per factor; NULL uses the study default.
beta	Latent regression coefficient or coefficient vector; NULL uses the design default.
seed	Optional random-number seed.

Value

The design catalog, or generated data and model metadata for the selected design.

convergence_sim_dejonckere_simple			
	<i>Convergence</i>	<i>simulation</i>	<i>design:</i>
	<i>gence_sim_dejonckere_simple</i>		<i>conver-</i>

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim_dejonckere_simple(n = 20L, variant = c("2025", "2022"), beta = NULL, seed = NULL)
```

Arguments

n	Number of observations to generate.
variant	Published design version, 2025 or 2022.
beta	Latent regression coefficient or coefficient vector; NULL uses the design default.
seed	Optional random-number seed.

Value

The design catalog, or generated data and model metadata for the selected design.

convergence_sim_ludtke_cfa

Convergence simulation design: convergence_sim_ludtke_cfa

Description

Provides a small reproducible convergence-study design. The catalog lists published design names and defaults; generators return data and model information without running a calibration study.

Usage

```
convergence_sim_ludtke_cfa(n = 30L, rho = 0.3, loading = 0.5, seed = NULL)
```

Arguments

<code>n</code>	Number of observations to generate.
<code>rho</code>	Latent factor correlation.
<code>loading</code>	Common standardized factor loading.
<code>seed</code>	Optional random-number seed.

Value

The design catalog, or generated data and model metadata for the selected design.

df_to_data

Prepare data: df_to_data

Description

Converts observations into native sample moments or raw-data input for the corresponding fitting route.

Usage

```
df_to_data(x, model, group = NULL, missing = c("listwise", "error"), scaling = c("n",
"n-1"))
```

Arguments

x	Input data frame; for mi_screen, a modification-index or score-test results table.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
group	Grouping-variable name or group identifiers, as appropriate for the data.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
scaling	Sample covariance divisor: n or n minus one.

Value

A prepared data list with observed-variable and group ordering.

<code>df_to_fcsem_data</code>	<i>Prepare data: df_to_fcsem_data</i>
-------------------------------	---------------------------------------

Description

Converts observations into native sample moments or raw-data input for the corresponding fitting route.

Usage

```
df_to_fcsem_data(x, model, missing = c("listwise", "error"), scaling = c("n", "n-1"))
```

Arguments

x	Input data frame; for mi_screen, a modification-index or score-test results table.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
scaling	Sample covariance divisor: n or n minus one.

Value

A prepared data list with observed-variable and group ordering.

df_to_fiml_data	<i>Prepare data: df_to_fiml_data</i>
-----------------	--------------------------------------

Description

Converts observations into native sample moments or raw-data input for the corresponding fitting route.

Usage

```
df_to_fiml_data(x, model, group = NULL)
```

Arguments

x	Input data frame; for mi_screen, a modification-index or score-test results table.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
group	Grouping-variable name or group identifiers, as appropriate for the data.

Value

A prepared data list with observed-variable and group ordering.

eqs_model	<i>Construct a model from EQS equations</i>
-----------	---

Description

Translate explicit EQS covariance-model sections into the shared magmaan model. Estimation and inference use magmaan, not the EQS application.

Usage

```
eqs_model(syntax, observed_names = NULL)
```

Arguments

syntax	One string containing /EQUATIONS, /VARIANCES and optionally /COVARIANCES. Unambiguous abbreviations with at least three letters, ! comments, ranges and an optional /END are accepted.
observed_names	Optional names in EQS data-column order. V1 maps to the first name, V2 to the second, etc. Without names, the data should have columns named V1, V2, etc.

Details

This initial subset supports single-group continuous covariance models. Every equation has one distinct error variable with a fixed unit coefficient: **E** for observed variables and **D** for factors. Every factor must have an observed indicator. Error numbering need not match equation numbering. A coefficient followed by ***** is free; a preceding number supplies its start. A coefficient without ***** is fixed (implicit coefficient one). Missing independent variances are free and missing covariances are zero. Identification follows explicit EQS specifications; no marker loading or single-indicator residual fix is added. Free parameters without supplied starts use magmaan's start policy, not EQS's internal start algorithm.

Job files, `/MODEL`, `/LABELS`, constraints, means (`V999`), multiple groups, shared errors, nonunit error paths and error/predictor or **E/D** covariances are rejected. Observed indicators participating in structural regressions are also outside this subset; separate observed predictors and purely observed path models are supported. Supply data and estimator separately. Scientific exponents must have a sign, e.g. `1e-3`, to distinguish `1E1`, which is a unit coefficient on error **E1**.

Value

A `magmaan_model_spec` for `fit_model()` and `prepare_model()`. `eqs_source` retains the original input; `syntax` is its explicit lavaan projection for rebuilding and refitting.

Examples

```
model <- eqs_model("
/EQUATIONS
V1 = *F1 + E1;
V2 = .7*F1 + E2;
V3 = .8*F1 + E3;
/VARIANCES
F1 = 1;
E1-E3 = .5*;
", observed_names = c("x1", "x2", "x3"))
```

est_change

Summarize case-rerun parameter changes

Description

Summarizes parameter changes from `case_rerun` under the selected standard-error convention.

Usage

```
est_change(rerun_out, parameters = NULL, se = c("standard", "robust.sem", "robust.huber.white"))
```

Arguments

<code>rerun_out</code>	Case-rerun result from <code>case_rerun</code> .
<code>parameters</code>	Optional parameter indices or names to retain.
<code>se</code>	Standard-error convention. <code>fit_model</code> is estimate-only and accepts only "none"; <code>SAM</code> and <code>est_change</code> expose the choices shown in their signatures.

Value

A parameter-change results table.

Note

The SE/type defaults standardize `SAM` or case-change diagnostics. They do not assert a generic misspecification-robust inference law; explicit robust or estimated-weight choices remain available within their supported scope.

<code>est_change_approx</code>	<i>Approximate standardized case influence (DF_{THETAS}) and gCD (one-step)</i>
--------------------------------	---

Description

The one-step approximation of `est_change()` without refitting: the approximate raw change $\Delta = (N/(N-1)) \cdot V \cdot s_i$ standardized by the full-sample SE, plus an approximate generalized Cook's distance $gcd_approx = \Delta' V_{sel}^{-1} \Delta$.

Usage

```
est_change_approx(
  fit,
  parameters = NULL,
  type = c("standard", "estimated.weight")
)
```

Arguments

<code>fit</code>	A fitted <code>magmaan_fit</code> (continuous ML/ULS/GLS) carrying raw data.
<code>parameters</code>	Optional parameter selector; default all free parameters.
<code>type</code>	"standard" (default; semfindr-style fixed-weight) or "estimated.weight" (misspecification-robust complete sandwich; estimators with a data-estimated weight: continuous GLS/WLS, ordinal DWLS/WLSMV, single- or multiple-group). Under "estimated.weight" both the one-step change and the SE / <code>gcd</code> metric use the complete-sandwich IJ covariance, the casewise dual of the estimated-weight standard error; see <code>est_change_raw_approx()</code> .

Details

This deliberately uses the *correct* finite-sample scaling, which differs from `semfindr::est_change_approx()` (v0.2.0) by two known constant factors: `semfindr` applies $N/(N-1)$ twice to `DFTHETAS` (one too many — the factor is already in the raw one-step change) and applies it only once inside `gCD` (one too few). With matched columns, `est_change_approx(magmaan)` relates to `semfindr` by `dftheta_magmaan = dftheta_semfindr · (N-1)/N` and `gcd_magmaan = gcd_semfindr · N/(N-1)`. Both factors are $O(1/N)$ and immaterial relative to the one-step error, but they have no first-principles basis (see the influence-function derivation; tracked as an upstream-PR item). The raw `est_change_raw_approx()` is unaffected (`semfindr` is correct there).

Value

A matrix with one column per selected parameter plus `gcd_approx`, one row per case.

Note

The SE/type defaults standardize SAM or case-change diagnostics. They do not assert a generic misspecification-robust inference law; explicit robust or estimated-weight choices remain available within their supported scope.

<code>est_change_raw</code>	<i>Case influence on raw parameter estimates (DFBETA / DFZ-THETA)</i>
-----------------------------	---

Description

For each case, (estimate with all cases) - (estimate without this case). With `standardized = FALSE` (default) these are raw coefficient changes; with `standardized = TRUE` they are changes in the standardized (std.all) solution. The change is NOT divided by a standard error (see `est_change()`). Mirror of `semfindr::est_change_raw()`.

Usage

```
est_change_raw(rerun_out, parameters = NULL, standardized = FALSE)
```

Arguments

<code>rerun_out</code>	Output of <code>case_rerun()</code> .
<code>parameters</code>	Optional selector (lavaan-syntax names and/or operators <code>=~</code> , <code>~</code> , <code>~~</code> , <code>~1</code>); default all free parameters.
<code>standardized</code>	Use the standardized (std.all) solution. Default <code>FALSE</code> .

Value

A matrix, one row per case, one column per selected parameter, `rownames = case ids`.

Note

The SE/type defaults standardize SAM or case-change diagnostics. They do not assert a generic misspecification-robust inference law; explicit robust or estimated-weight choices remain available within their supported scope.

est_change_raw_approx

Approximate case influence on raw parameter estimates (one-step, no refit)

Description

$(N/(N-1)) \cdot V \cdot s_i$ per case, the one-step approximation of `est_change_raw()` without re-fitting. Mirror of `semfindr::est_change_raw_approx()`.

Usage

```
est_change_raw_approx(
  fit,
  parameters = NULL,
  type = c("standard", "estimated.weight")
)
```

Arguments

fit	A fitted <code>magmaan_fit</code> carrying raw data ("standard": continuous ML/ULS/GLS; "estimated.weight": continuous GLS/WLS/ULS or ordinal DWLS/WLSMV/ULSMV).
parameters	Optional parameter selector; default all free parameters.
type	"standard" (default; semfindr-style fixed-weight) or "estimated.weight" (misspecification-robust complete sandwich; estimators with a data-estimated weight, single- or multiple-group).

Details

With `type = "estimated.weight"` this is the misspecification-robust ("complete-sandwich") one-step change for a moment-quadratic fit whose weight is estimated from the data (continuous GLS/WLS, ordinal DWLS/WLSMV; ULS is fixed-weight and so unchanged): the per-case influence carries the data-dependent-weight term that `semfindr` / Pek & MacCallum drop by treating the estimator weight as fixed. That term is $O_p(N^{-1})$ under a correct model but $O_p(N^{-1/2})$ under misspecification (Hall-Inoue order promotion), so the two regimes agree at the null and diverge under misfit; it is the casewise dual of the estimated-weight standard error. The result then carries two attributes: "naive" (the fixed-weight one-step change) and "weight_diagnostic" (their difference, the per-case $\Delta'W_d$ term). There is no `semfindr` counterpart; this is a frontier extension.

Both regimes support single- and multiple-group fits; multiple-group rows are block-stacked (`g{b}_{row}` ids, group-suffixed columns).

Value

A matrix, one row per case, one column per selected parameter (plus the "naive" / "weight_diagnostic" attributes when `type = "estimated.weight"`).

Note

The SE/type defaults standardize SAM or case-change diagnostics. They do not assert a generic misspecification-robust inference law; explicit robust or estimated-weight choices remain available within their supported scope.

`factor_score_precision`

Factor-score precision diagnostics

Description

Computes precision diagnostics for factor scores from a fit and complete raw observations.

Usage

```
factor_score_precision(fit, data)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Value

Native factor-score precision diagnostics.

`factor_scores`

Compute factor scores

Description

Computes factor scores from raw observations. The default is regression for continuous fits and EBM for ordinal fits; explicit choices include regression, bartlett, EBM, ML and EAP.

Usage

```
factor_scores(fit, data, method = NULL)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>method</code>	Method selector; see the available choices in the usage signature and details.

Value

Factor-score estimates returned by the native scoring operation.

<code>fcsem_fit_measures</code>	<i>Factor-component SEM: fcsem_fit_measures</i>
---------------------------------	---

Description

Constructs, fits or summarizes the native factor-component SEM representation. These objects use their own fitting route.

Usage

```
fcsem_fit_measures(fit)
```

Arguments

<code>fit</code>	A fitted magmaan model.
------------------	-------------------------

Value

A factor-component model specification, fitted model or named post-fit summary, according to the operation.

<code>fcsem_model_spec</code>	<i>Factor-component SEM: fcsem_model_spec</i>
-------------------------------	---

Description

Constructs, fits or summarizes the native factor-component SEM representation. These objects use their own fitting route.

Usage

```
fcsem_model_spec(syntax, ...)
```

Arguments

<code>syntax</code>	A single model-syntax string.
<code>...</code>	Additional options forwarded to model construction or the underlying named operation.

Value

A factor-component model specification, fitted model or named post-fit summary, according to the operation.

`fcsem_standard_errors`

Factor-component SEM: fcsem_standard_errors

Description

Constructs, fits or summarizes the native factor-component SEM representation. These objects use their own fitting route.

Usage

```
fcsem_standard_errors(fit)
```

Arguments

<code>fit</code>	A fitted magmaan model.
------------------	-------------------------

Value

A factor-component model specification, fitted model or named post-fit summary, according to the operation.

`fcsem_standardized_rows`

Factor-component SEM: fcsem_standardized_rows

Description

Constructs, fits or summarizes the native factor-component SEM representation. These objects use their own fitting route.

Usage

```
fcsem_standardized_rows(fit, vcov = NULL)
```

Arguments

fit	A fitted magmaan model.
vcov	Explicit covariance matrix of free parameters, in theta order.

Value

A factor-component model specification, fitted model or named post-fit summary, according to the operation.

fit_index_intervals_misspec

Candidate continuous misspecification-robust fit-index intervals

Description

Lab-only Wald intervals for continuous complete-data ML and FIML, including independent groups, random x and affine equality constraints. Points agree with `policy_fit_measures()`. The variance candidate has not been adopted or coverage-calibrated.

Usage

```
fit_index_intervals_misspec(fit, level = 0.90)
```

Arguments

fit	A fitted magmaanlab model retaining raw observations.
level	Confidence level strictly between zero and one, default 0.90.

Details

The casewise likelihood contrast gives the first-order user discrepancy variance V1. The second-order V2 is $2\text{tr}(M^2)/N^2$, where $M = (J_s^{-1} - \Delta J_0^{-1} \Delta^\top) K_s$ uses saturated observed information, constraint-reduced model observed information, the implied-moment Jacobian and the saturated score outer product. Covariance-only ML includes its profiled group-mean nuisance parameters.

`wald_f` maps a Wald interval for the corrected discrepancy to RMSEA. `wald_sqrt` uses its square-root coordinate and requires a positive corrected discrepancy. CFI and TLI use the joint user/baseline first-order delta method plus V2 for the user discrepancy; their intervals coincide for both variants. CFI endpoints are clipped to [0,1]; TLI endpoints are unbounded. RMSEA lower bounds are clipped at zero. Ordinal, ML2S, other estimators, clustered, penalized and failed fits return typed unavailable intervals.

Value

A data frame with index, variant, estimate, lower, upper, V1, V2 and reason. V1/V2 are always on the user discrepancy scale, before mapping to an index. The details attribute includes casewise and fit discrepancies, their corrected point, Takeuchi and quadratic traces, sample size and level. Unavailable intervals have NA bounds and explicit reasons.

See Also

[policy_fit_measures](#)

<code>fit_measures</code>	<i>Normal-theory fit measures, optionally with FMG robust p-values.</i>
---------------------------	---

Description

Convenience wrapper over the existing fit-measure primitives. By default it reports the standard ML chi-square, df, p-value, baseline chi-square, and ordinary fit measures. Categorical models report the native `n F` statistic; lavaan's `(n - G) F` standard reporting is available through `convention_inference()`. Passing `fmg = TRUE` or a character vector of FMG test labels adds an `$fmg` diagnostic table from `fmg_tests()`.

Usage

```
fit_measures(fit, baseline = NULL, fmg = NULL, robust = NULL, data = NULL)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>baseline</code>	Optional baseline result from <code>magmaan_core\$measures_baseline()</code> .
<code>fmg</code>	NULL/FALSE for no FMG table, TRUE for default FMG tests, or a character vector of FMG test names.
<code>robust</code>	Optional scalar list with lavaan-style scaled user and baseline test fields. Automatic robust estimator dispatch is not yet wired.
<code>data</code>	Optional complete raw data used only when FMG is requested and the fit does not carry <code>\$raw_data</code> .

<code>fit_measures_change</code>	<i>Case influence on fit measures</i>
----------------------------------	---------------------------------------

Description

For each case, (fit measure with all cases) - (fit measure without this case) for each requested measure. Mirror of `semfindr::fit_measures_change()`.

Usage

```
fit_measures_change(
  rerun_out,
  fit_measures = c("chisq", "cfi", "rmsea", "tli")
)
```

Arguments

rerun_out Output of `case_rerun()`.

fit_measures Names of fit measures (as returned by `fit_measures()`); default `c("chisq", "cfi", "rmsea", "tli")`.

Value

A matrix, one row per case, one column per measure, rownames = case ids.

fit_measures_misspec	<i>Misspecification-robust ordinal fit measures (estimated-weight intervals)</i>
-----------------------------	--

Description

Estimated-weight (categorical DWLS) misspecification inference for the absolute and incremental fit indices RMSEA, CRMR, SRMR, CFI and TLI, each with a confidence interval that propagates the sampling variability of the estimated polychoric weight (the gamma channel). The single consolidated surface over the per-index entry points. Single- and multi-group all-ordinal DWLS fits only. Exact residual inference includes pooled weights and cross-group parameter influence; the OPG default keeps its legacy blockwise approximation. Residual point fields in this consolidated table remain uncorrected.

Usage

```
fit_measures_misspec(
  fit,
  ordinal_stats = NULL,
  estimated_weight = TRUE,
  conf_level = 0.9,
  eig_tol = 1e-10,
  first_stage = c("opg", "exact")
)
```

Arguments

fit A fitted all-ordinal DWLS magmaan object.

ordinal_stats The categorical sample statistics used for the fit, with `moment_influence` and `int_data` (e.g. from `data_ordinal_stats_from_df()`). Required and passed explicitly, like `robust_ordinal()` and the profile bindings: a fitted object does not retain the integer data the estimated-weight inference needs.

estimated_weight `TRUE` (the misspecification-robust default) propagates the polychoric-weight sampling variability; `FALSE` is the fixed-weight comparator.

conf_level Two-sided confidence level for the intervals.

<code>eig_tol</code>	Eigenvalue tolerance for the profile-contrast spectrum.
<code>first_stage</code>	OPG (default lab convention) or exact empirical sampling influence. Exact requires complete raw data or supplied exact rows; fitting weights remain the NACOV weights used for estimation.

Details

CFI carries a trustworthy interval and is largely robust to weight estimation; TLI's point is calibrated but its interval is conservative and unreliable at strong misfit (the generalized-df ratio is ill-conditioned). See `cfi_tli_misspec_inference.tex`.

Value

A named list: `rmsea/crmr/srmr/cfi/tli` with `.ci.lower` and `.ci.upper`, the exact-fit `rmsea.pvalue` and `crm.pvalue`, the user and baseline `chisq/df`, `conf.level`, `estimated.weight`, and `warnings`.

`fit_measures_misspec_mixed_ordinal`

Misspecification-robust mixed continuous/ordinal DWLS fit measures

Description

Exact discrepancy points also include the known N-divisor bias of the continuous covariance moments. Polychoric/polyserial bias remains omitted. Exact residual inference uses the full pooled influence map; OPG retains its legacy blockwise approximation and defaults. The consolidated residual point fields remain uncorrected.

Usage

```
fit_measures_misspec_mixed_ordinal(
  fit,
  mixed_stats = NULL,
  estimated_weight = TRUE,
  conf_level = 0.9,
  eig_tol = 1e-10,
  first_stage = c("opg", "exact")
)
```

Arguments

<code>fit</code>	A mixed continuous/ordinal DWLS fit returned by the core mixed ordinal fitting helpers.
<code>mixed_stats</code>	The mixed continuous/ordinal sample statistics used for the fit, with <code>moment_influence</code> and either <code>gamma_diag_influence</code> or raw data available (e.g. from <code>data_mixed_ordinal_stats_from_df()</code>).

<code>estimated_weight</code>	TRUE (the misspecification-robust default) propagates the mixed DWLS weight sampling variability; FALSE is the fixed-weight comparator.
<code>conf_level</code>	Two-sided confidence level for the intervals.
<code>eig_tol</code>	Eigenvalue tolerance for the profile-contrast spectrum.
<code>first_stage</code>	OPG (default lab convention) or exact empirical sampling influence. Exact requires complete raw data or supplied exact rows; fitting weights remain the NACOV weights used for estimation.

Value

A named list with the same fields as `fit_measures_misspec()`.

<code>fit_measures_noniterative</code>	<i>Fit indices (CFI/TLI/RMSEA + SRMR) for a non-iterative CFA fit.</i>
--	--

Description

The user statistic is the residual goodness-of-fit T on df degrees of freedom under the chosen discrepancy, referred to an independence-model baseline under the SAME discrepancy. Naive indices treat T as central chi-square; robust indices use the Satorra-Bentler scaling `scale_c` (user) and the baseline scaling (independence model) to correct the mixture reference. `logl/unrestricted.logl/aic/bic/bic2` are reported as NA: these are likelihood information criteria and are not defined for the closed-form non-ML estimator.

Usage

```
fit_measures_noniterative(
  fit,
  discrepancy = c("ntml", "uls"),
  gamma = c("nt", "empirical"),
  data = NULL,
  baseline = NULL,
  fmg = NULL
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>discrepancy</code>	"ntml" (model-implied normal-theory) or "uls".
<code>gamma</code>	"nt" (normal-theory) or "empirical" (needs data).
<code>data</code>	Raw data, required when <code>gamma = "empirical"</code> .
<code>baseline</code>	Optional precomputed <code>list(chi2=, df=)</code> ; default is the same-discrepancy independence model.
<code>fmg</code>	Optional passthrough to <code>fmg_tests()</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

fit_ml_fcsem

Factor-component SEM: fit_ml_fcsem

Description

Constructs, fits or summarizes the native factor-component SEM representation. These objects use their own fitting route.

Usage

```
fit_ml_fcsem(model, data, missing = c("listwise", "error"), control = NULL)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.

Value

A factor-component model specification, fitted model or named post-fit summary, according to the operation.

fit_model

Estimate a structural equation model

Description

Fits the requested objective and returns estimates and diagnostics. Inference is composed explicitly after fitting.

Usage

```
fit_model(model, data, estimator = "ML", groups = NULL, ..., cluster = NULL, ordered = NULL,
  parameterization = "delta", missing = c("listwise", "error", "pairwise"), pd_gamma = c("over",
    "nominal"), se = "none", test = "none", W = NULL, optimizer = NULL, control = NULL,
  bounds = NULL, stage2_weight = "nt", dls_a = 0.5, stage1_regularization = NULL,
  psd = FALSE, covariance = NULL, barrier = NULL, weight = NULL, options = NULL)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
estimator	Estimation objective. Supported objectives depend on the data kind and fitting route.
groups	Grouping-variable name used when constructing a model from syntax.
...	Additional options forwarded to model construction or the underlying named operation.
cluster	Cluster-variable name or cluster identifiers for two-level fitting.
ordered	Names of ordinal observed variables.
parameterization	Ordinal model parameterization; currently delta.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
pd_gamma	Pairwise fourth-moment covariance convention: overlap or nominal.
se	Standard-error convention. fit_model is estimate-only and accepts only "none"; SAM and est_change expose the choices shown in their signatures.
test	Test convention. fit_model is estimate-only and accepts only "none".
W	Fixed fitting-weight matrix, or a per-group list in model moment order.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts. coordinate_scaling accepts "none", "sample_units", "information" (default), "objective", and "objective_downward". The last two use the actual fixed-weight least-squares Gauss–Newton diagonal at the start, with sample-unit multipliers clamped to [1e-6, 1e6] or [1e-3, 1], respectively. Invalid or nonpositive diagonal entries retain the sample unit. They support continuous fixed-weight LS and ML2S Stage-2 LS with scalar backends; ML, direct FIML, residual backends, and ordinal/mixed LS routes reject them as Unsupported. Ordinal/mixed sample-unit and information modes retain their existing behavior.
bounds	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
stage2_weight	Two-stage fitting weight selector, such as nt or adf.
dls_a	DLS mixing scalar in [0, 1]: zero is normal theory and one is ADF.
stage1_regularization	Optional Stage-1 regularization specification for ML2S.
psd	Whether to constrain model covariance blocks to be positive semidefinite.
covariance	Covariance-domain policy: unrestricted, psd or barrier.
barrier	Barrier options with target and non-negative weight.

weight	Weight selector or explicit fitting weight as required by the route.
options	Named fitting-options list: preset, starts, optimizer, convergence and marker. The lavaan-0.7.2 preset selects all four components. marker = "default" keeps requested markers; marker = "lavaan-0.7.2" applies the weak-marker rule for continuous ML, FIML and ordinal/mixed DWLS and retries the original model if the switched fit is rejected. The fit retains the actual model and fitting\$marker_switch (lv, old, new, signed r_old and r_new rounded to three decimals, reverted). The rule reads each route's saturated H1 covariance. Conditional-x residual H1 and composite adaptation are unsupported. Nested inference and refit_from_null explicitly refuse switched marker coordinates. Components may also be supplied in control\$fitting_options.

Value

A magmaan_fit list with parameter estimates, fitting options, convergence and admissibility diagnostics.

Deletion provenance

Dataframe helpers, prepared data and fitted lab objects retain a read-only **deletion** record. Its **policy** identifies listwise, error, FIML or pairwise row handling; **known** indicates whether supplied-row counts are available. **per_group** contains **group**, **rows_supplied**, **rows_used**, **rows_deleted**, and exclusive reason counts **missing_model_values**, **missing_group_label**, **unknown_group_label**, and **missing_cluster_id**. Reasons take precedence in that order: group labels, cluster IDs, model values. Rows without an assignable group appear in a separate NA group row. Single-level routes still refuse missing or unknown group labels; two-level listwise preparation deletes missing group or cluster IDs. FIML and ML2S retain incomplete rows without listwise deletion; existing native refusals remain. Summary-only inputs without retained provenance report unknown supplied/deleted counts, rather than claiming that no earlier deletion occurred.

fit_noniterative_cfa *Fit a non-iterative CFA estimator.*

Description

Fit a non-iterative CFA estimator.

Usage

```
fit_noniterative_cfa(
  partable,
  sample_stats,
  estimator = "guttman_lavaan",
  composite = "auto",
  admissibility = "raw",
```

```

margin = 1e-04,
beta0 = 1,
rate = 0.5,
score_conditioning = "raw",
score_floor0 = 1,
score_rate = 0.5,
h_conditioning = "raw",
h_floor0 = 1,
h_rate = 0.5
)

```

Arguments

partable	A magmaan/lavaan parameter table.
sample_stats	A magmaan sample-statistics list (S, nobs, ...).
estimator	Non-iterative estimator. "guttman_lavaan" keeps the legacy lavaan-like Spearman/incidence Guttman map (AR communality); "guttman_aligned" uses the block-GLS H diagonal and aligned score reconstruction. Old names "guttman" and "guttman_gls_aligned" are still accepted as aliases.
composite	Composite weights used by the Guttman regression: "auto" preserves the estimator default, "unit" uses incidence weights, and "standardized" uses incidence weights after indicator standardization.
admissibility	Communality admissibility policy: "raw" leaves the closed-form estimate unchanged, "hard" clips it to the symmetric box, and "soft" applies the smooth box map.
margin	Symmetric communality-box margin.
beta0	Base soft-clamp sharpness.
rate	Sample-size exponent used to resolve soft-clamp sharpness as $\text{beta0} * \text{nobs}^{\text{rate}}$.
score_conditioning	Score-covariance conditioning policy. "raw" preserves the unconditioned map, "hard" applies the exact normalized eigenvalue floor, and "soft" applies its smooth spectral analogue. Non-raw policies are supported for aligned unit and standardized composites.
score_floor0	Base normalized score-eigenvalue floor.
score_rate	Sample-size exponent resolving the floor as $\text{score_floor0} * \text{nobs}^{(-\text{score_rate})}$; the result must lie strictly in (0, 1).
h_conditioning	Feasibility-only fixed-diagonal PSD repair for the aligned H proxy: "raw", "hard", or "soft". Non-raw modes currently support point estimation only.
h_floor0	Base normalized H-eigenvalue floor.
h_rate	Sample-size exponent resolving the H floor as $\text{h_floor0} * \text{nobs}^{(-\text{h_rate})}$.

Value

A magmaan fit object (same shape as `fit_model()`), usable by the inference helpers below and by partable inspection.

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

`fit_noniterative_cfa_metric`

Fit the estimator-side metric-constrained non-iterative CFA map.

Description

This is not the inference-side Omega projection returned by `noniterative_cfa_constrained()`. It imposes the metric loading-shape constraint inside the Sigma/H reconstruction itself: a common standardized loading shape is estimated across groups and then converted back to the marker chart represented by `partable`.

Usage

```
fit_noniterative_cfa_metric(
  partable,
  sample_stats,
  estimator = "guttman_aligned",
  composite = "auto",
  admissibility = "raw",
  margin = 1e-04,
  beta0 = 1,
  rate = 0.5,
  score_conditioning = "raw",
  score_floor0 = 1,
  score_rate = 0.5,
  h_conditioning = "raw",
  h_floor0 = 1,
  h_rate = 0.5
)
```

Arguments

<code>partable</code>	A magmaan/lavaan parameter table.
<code>sample_stats</code>	A magmaan sample-statistics list (S, nobs, ...).
<code>estimator</code>	Non-iterative estimator. "guttman_lavaan" keeps the legacy lavaan-like Spearman/incidence Guttman map (AR communality); "guttman_aligned" uses the block-GLS H diagonal and aligned score reconstruction. Old names "guttman" and "guttman_gls_aligned" are still accepted as aliases.

<code>composite</code>	Composite weights used by the Guttman regression: "auto" preserves the estimator default, "unit" uses incidence weights, and "standardized" uses incidence weights after indicator standardization.
<code>admissibility</code>	Communality admissibility policy: "raw" leaves the closed-form estimate unchanged, "hard" clips it to the symmetric box, and "soft" applies the smooth box map.
<code>margin</code>	Symmetric communality-box margin.
<code>beta0</code>	Base soft-clamp sharpness.
<code>rate</code>	Sample-size exponent used to resolve soft-clamp sharpness as $\text{beta0} * \text{nobs}^{\text{rate}}$.
<code>score_conditioning</code>	Score-covariance conditioning policy. "raw" preserves the unconditioned map, "hard" applies the exact normalized eigenvalue floor, and "soft" applies its smooth spectral analogue. Non-raw policies are supported for aligned unit and standardized composites.
<code>score_floor0</code>	Base normalized score-eigenvalue floor.
<code>score_rate</code>	Sample-size exponent resolving the floor as $\text{score_floor0} * \text{nobs}^{(-\text{score_rate})}$; the result must lie strictly in (0, 1).
<code>h_conditioning</code>	Feasibility-only fixed-diagonal PSD repair for the aligned H proxy: "raw", "hard", or "soft". Non-raw modes currently support point estimation only.
<code>h_floor0</code>	Base normalized H-eigenvalue floor.
<code>h_rate</code>	Sample-size exponent resolving the H floor as $\text{h_floor0} * \text{nobs}^{(-\text{h_rate})}$.

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

`fit_noniterative_cfa_restricted`

Fit the residual/loading restricted non-iterative CFA map.

Description

Residual-variance equality constraints are imposed in the Guttman H communality step before the loadings are reconstructed. Loading-only linear equalities are then projected with a Sigma-only composite metric. Rows that mix residual and loading parameters, or rows involving factor variances, intercepts, or latent means, are rejected.

Usage

```
fit_noniterative_cfa_restricted(
  partable,
  sample_stats,
  estimator = "guttman_aligned",
  communality = "triad_wls",
  composite = "auto",
  admissibility = "raw",
  margin = 1e-04,
  beta0 = 1,
  rate = 0.5,
  score_conditioning = "raw",
  score_floor0 = 1,
  score_rate = 0.5,
  h_conditioning = "raw",
  h_floor0 = 1,
  h_rate = 0.5
)
```

Arguments

partable	A magmaan/lavaan parameter table.
sample_stats	A magmaan sample-statistics list (S, nobs, ...).
estimator	Non-iterative estimator. "guttman_lavaan" keeps the legacy lavaan-like Spearman/incidence Guttman map (AR communality); "guttman_aligned" uses the block-GLS H diagonal and aligned score reconstruction. Old names "guttman" and "guttman_gls_aligned" are still accepted as aliases.
communality	Least-squares-form communality rule for the residual/H step: "triad_wls" (default), "triad_wls_joint", "triad_ls", or "extended_triad_ls". Old names "gmm_block", "gmm_full", and "anchor_triad_ls" are still accepted.
composite	Composite weights used by the Guttman regression: "auto" preserves the estimator default, "unit" uses incidence weights, and "standardized" uses incidence weights after indicator standardization.
admissibility	Communality admissibility policy: "raw" leaves the closed-form estimate unchanged, "hard" clips it to the symmetric box, and "soft" applies the smooth box map.
margin	Symmetric communality-box margin.
beta0	Base soft-clamp sharpness.
rate	Sample-size exponent used to resolve soft-clamp sharpness as $\text{beta0} * \text{nobs}^{\text{rate}}$.
score_conditioning	Score-covariance conditioning policy. "raw" preserves the unconditioned map, "hard" applies the exact normalized eigenvalue floor, and "soft" applies its smooth spectral analogue. Non-raw policies are supported for aligned unit and standardized composites.

<code>score_floor0</code>	Base normalized score-eigenvalue floor.
<code>score_rate</code>	Sample-size exponent resolving the floor as <code>score_floor0 * nobs^(-score_rate)</code> ; the result must lie strictly in (0, 1).
<code>h_conditioning</code>	Feasibility-only fixed-diagonal PSD repair for the aligned H proxy: "raw", "hard", or "soft". Non-raw modes currently support point estimation only.
<code>h_floor0</code>	Base normalized H-eigenvalue floor.
<code>h_rate</code>	Sample-size exponent resolving the H floor as <code>h_floor0 * nobs^(-h_rate)</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

<code>fit_twolevel</code>	<i>Fit a two-level structural equation model</i>
---------------------------	--

Description

Fits the native two-level model using cluster identifiers and optional group identifiers.

Usage

```
fit_twolevel(model, data, cluster, group = NULL, optimizer = "nlopt-lbfgs", control = NULL,
             bounds = NULL, missing = c("listwise", "error"))
```

Arguments

<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>cluster</code>	Cluster-variable name or cluster identifiers for two-level fitting.
<code>group</code>	Grouping-variable name or group identifiers, as appropriate for the data.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>bounds</code>	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
<code>missing</code>	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.

Value

A two-level fitted-model result.

fmg_nested	<i>Foldnes-Moss-Gronneberg diagnostics for a nested continuous/FIML/ML2S pair.</i>
------------	--

Description

Applies the FMG eigenvalue-tail transforms to the Satorra-2000 restriction-map difference spectrum returned by `robust_nested_lrt()`. For complete-data ML pairs an explicit `_rls` suffix uses the difference between the two Browne RLS residual statistics, evaluated at the ML estimates, as the asymptotically equivalent base statistic. An `_ug` suffix selects the Browne/Du-Bentler finite-sample correction to the nested Gamma spectrum. Unsuffixes tests and `_ml` retain the likelihood-ratio difference. FIML and two-stage ML (ML2S) support only the biased Gamma and ML base.

Usage

```
fmg_nested(
  fit_H1,
  fit_H0,
  data = NULL,
  tests = NULL,
  A.method = c("exact", "delta"),
  weight = NULL,
  gamma = NULL
)
```

Arguments

<code>fit_H1</code>	Less-restricted fitted model.
<code>fit_H0</code>	More-restricted fitted model.
<code>data</code>	Raw complete data for complete-data ML or continuous LS pairs. FIML and ML2S pairs use <code>fit_H1\$raw_data</code> and reject <code>data</code> , matching <code>robust_nested_lrt()</code> .
<code>tests</code>	Character vector of semTests-style test names, or NULL for the nested defaults (SB, pEBA2, pEBA4, pEBA6, pOLS). Complete-data pairs accept explicit <code>_ml</code> and <code>_rls</code> bases; unsuffixes names use ML.
<code>A.method</code>	"exact" (default) or "delta".
<code>weight</code>	Explicit fitting weight for continuous WLS. Continuous ULS and GLS rebuild their identity/normal-theory weights.
<code>gamma</code>	Continuous-LS Gamma source. NULL chooses normal theory for ULS (matching lavaan's <code>robust.sem.nt</code> default) and empirical Gamma for GLS/WLS. Otherwise use "empirical" or "normal".

Value

A `magmaan_fmg_tests` data frame. Numerical diagnostics from the nested-test backend are available in `attr(x, "warnings")`.

Note

This is an FMG/semTests replication-compatibility route. Its NT/fixed-weight conventions reproduce the documented comparator and are distinct from the generic nested policy in `robust_nested_lrt()`.

<code>fmg_nested_ordinal</code>	<i>Foldnes-Moss-Gronneberg eigenvalue-tail diagnostics for a nested ordinal pair.</i>
---------------------------------	---

Description

The nested-model analogue of `fmg_tests_ordinal()`. The Satorra-2000 reduction of the two polychoric least-squares fits yields a difference triple — the unscaled LS difference statistic $T_{H0} - T_{H1}$, the restriction rank (`df_diff`), and the difference spectrum (the nested UGamma eigenvalues) — and the same FMG eigenvalue-tail transforms are applied to it, exactly as `fmg_tests_ordinal()` applies them to a single-model UGamma spectrum.

Usage

```
fmg_nested_ordinal(
  fit_H1,
  fit_H0,
  ordinal_stats,
  tests = NULL,
  weight = "",
  A.method = c("delta", "exact")
)

fmg_nested_mixed_ordinal(
  fit_H1,
  fit_H0,
  mixed_stats,
  tests = NULL,
  weight = "",
  A.method = c("delta", "exact")
)
```

Arguments

<code>fit_H1</code>	Less-restricted (e.g. configural) ordinal least-squares fit.
<code>fit_H0</code>	More-restricted (e.g. metric / <code>group.equal</code>) ordinal fit.
<code>ordinal_stats</code>	The categorical sample statistics used for both fits (the object from <code>data_ordinal_stats_from_d</code> as in <code>fmg_tests_ordinal()</code>).
<code>tests, weight</code>	As in <code>fmg_tests_ordinal()</code> .
<code>A.method</code>	"delta" (default) or "exact".
<code>mixed_stats</code>	The mixed continuous/ordinal sample statistics used for both fits (the object from <code>data_mixed_ordinal_stats_from_df()</code>).

Details

`A.method` defaults to `"delta"`: the configural-to-metric ordinal invariance step is non-nested (Wu-Estabrook metric frees the group-2+ scale/intercepts that configural fixes), so the exact parameter-nesting restriction does not apply — the moment-Jacobian column-space (`delta`) restriction is what lavaan's `lavTestLRT(method = "satorra.2000")` uses there. (`robust_nested_lrt()` keeps the `"exact"` default for nested complete-data/FIML pairs.)

Value

A `magmaan_fmg_tests` data frame, identical in shape to `fmg_tests_ordinal()`. Numerical diagnostics from the nested-test backend are available in `attr(x, "warnings")`.

<code>fmg_pvalues</code>	<i>Foldnes-Moss-Gronneberg goodness-of-fit p-values for a fitted magmaan model.</i>
--------------------------	---

Description

Back-compatible named-vector view over `fmg_tests()`. New code that needs diagnostics should call `fmg_tests()` or `fit_measures(..., fmg = tests)`.

Usage

```
fmg_pvalues(
  fit,
  data = NULL,
  tests = NULL,
  weight = NULL,
  gamma = c("empirical", "normal")
)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>data</code>	Optional complete raw data. If omitted, <code>fit\$raw_data</code> is used.
<code>tests</code>	Character vector of semTests-style test names. Recognised types: <code>std</code> , <code>sb</code> , <code>ss</code> , <code>mv</code> , <code>sf</code> , <code>all</code> , <code>pall</code> , <code>eba<j></code> , <code>peba<j></code> , <code>pols<gamma></code> , each optionally suffixed <code>_ug</code> (unbiased Gamma-hat) and <code>_ml</code> / <code>_rls</code> (base statistic; default <code>rls</code>).
<code>weight</code>	Optional explicit continuous-WLS fitting weight, passed to <code>fmg_tests()</code> .
<code>gamma</code>	Continuous-LS moment-covariance source, passed to <code>fmg_tests()</code> .

Value

A named numeric vector of p-values; names are the canonical test labels (e.g. `peba4_rls`, `sb_ug_rls`), matching `semTests::pvalues()`.

fmg_tests

Foldnes-Moss-Gronneberg goodness-of-fit diagnostics.

Description

Computes FMG p-values and diagnostics for a complete-data ML or continuous ULS/GLS/WLS fit, or a FIML (missing-data) fit. Complete-data fits must carry complete raw data, as fits from `fit_model(..., data.frame)` do, or callers can pass complete raw `data` explicitly.

Usage

```
fmg_tests(
  fit,
  tests = NULL,
  data = NULL,
  weight = NULL,
  gamma = c("empirical", "normal")
)
```

Arguments

fit	A fitted magmaan ML, continuous ULS/GLS/WLS, FIML, or ML2S model.
tests	Character vector of semTests-style test names, or <code>NULL</code> for the recommended defaults (complete-data or FIML-appropriate). Recognised types: <code>std</code> , <code>sb</code> , <code>ss</code> , <code>mv</code> , <code>sf</code> , <code>all</code> , <code>pall</code> , <code>eba<j></code> , <code>peba<j></code> , <code>pols<gamma></code> , each optionally suffixed <code>_ug</code> and <code>_ml</code> / <code>_rls</code> ; availability depends on the estimator as described above.
data	Optional complete raw data (complete-data fits only). Usually unnecessary for new fits that retain <code>\$raw_data</code> .
weight	The explicit fitting weight for a continuous WLS fit. Ignored for ML, ULS, and GLS. Magmaan fit objects do not retain caller-supplied WLS weights.
gamma	For continuous ULS/GLS/WLS fits, either <code>"empirical"</code> (the default raw-data fourth-moment covariance) or <code>"normal"</code> (the normal-theory moment covariance at the fitted sample covariance).

Details

FIML fits (`fit_fiml()` / `fit_model(..., estimator = "FIML")`), single- or multi-group) are supported: the missing-data UGamma spectrum is computed first-principles from the saturated-model EM ACOV and either saturated or model-implied H1 information, with the FIML LRT as the base statistic. Under FIML only the biased Gamma-hat and the ML base are defined, so `_ug` and `_rls` are rejected (an unsuffixed base resolves to ML). This is a principled construction, not a port of semTests' (unsound) FIML handling.

Two-stage ML (ML2S) fits (`fit_ml2s()` / `fit_model(..., estimator = "ML2S")`) are supported the same way: the df-dimensional UGamma spectrum and the Stage-2 ML base chi-square are taken from the two-stage inference already attached to the fit (`fit$ml2s`). Unsuffixed tests use that ML base; an explicit `_rls` suffix instead uses the full fitted-moment RLS residual quadratic, including restricted means, with the same spectrum. As under FIML, `_ug` is rejected.

Value

A data frame with scalar diagnostics plus list-columns for the raw UGamma spectrum and the method-specific lambda vectors.

<code>fmg_tests_ordinal</code>	<i>Foldnes-Moss-Gronneberg goodness-of-fit diagnostics for an ordinal fit.</i>
--------------------------------	--

Description

Applies the FMG eigenvalue-tail transforms to an ordinal (all-categorical) or mixed continuous/ordinal least-squares fit (DWLS/WLS/ULS over polychoric and polyserial moments). The UGamma spectrum, base chi-square, and df come from the same polychoric-NACOV sandwich as `robust_ordinal()` / `robust_mixed_ordinal()`, so the categorical stats are supplied explicitly the same way. Single- and multi-group fits are supported.

Usage

```
fmg_tests_ordinal(fit, ordinal_stats, tests = NULL, weight = "")

fmg_tests_mixed_ordinal(fit, mixed_stats, tests = NULL, weight = "")
```

Arguments

<code>fit</code>	A fitted magmaan ordinal (or mixed-ordinal) least-squares model.
<code>ordinal_stats, mixed_stats</code>	The categorical sample statistics built by <code>magmaan_core\$data_ordinal_stats_from_df()</code> / <code>data_mixed_ordinal_stats_from_df()</code> (the same object passed to <code>robust_ordinal()</code>).
<code>tests</code>	Character vector of semTests-style test names, or NULL for the ordinal defaults (SB, pEBA2, pEBA4, pEBA6, pOLS).
<code>weight</code>	Estimation weight ("DWLS", "WLS", "ULS"); empty resolves from <code>fit\$estimator</code> , matching <code>robust_ordinal()</code> .

Details

An ordinal LS fit has one base statistic, the LS chi-square $N F_{\min}$; there is no ML/RLS split and no Du-Bentler unbiased Gamma, so test names suffixed `_ml` or `_ug` are rejected. As on the complete-data and FIML paths, the pEBA/pOLS/PALL transforms are magmaan constructions with no external oracle.

Value

A magmaan_fmg_tests data frame, identical in shape to fmg_tests().

```
frontier_fit_fiml_multiinfo
```

Experimental fitting route: frontier_fit_fiml_multiinfo

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_fiml_multiinfo(model, data, eta = 1.25, weight = NULL, optimizer = "port", control = NULL,
  bounds = NULL, target = c("joint", "determinacy"))
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
eta	Compatibility barrier parameter; used when weight is omitted.
weight	Weight selector or explicit fitting weight as required by the route.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.
bounds	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
target	Barrier target: joint covariance or factor-score determinacy.

Value

A fitted-model list with parameter estimates and route diagnostics.

frontier_fit_fiml_psd
Experimental fitting route: frontier_fit_fiml_psd

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_fiml_psd(model, data, optimizer = "nlopt-slsqp", control = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, psd_factor = c("triangular", "square"))
```

Arguments

psd_factor	Covariance factor: triangular (default) or square.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.
start_eigen_floor	Eigenvalue floor used to construct a feasible covariance start.
feasibility_tol	Numerical tolerance for covariance feasibility.

Value

A fitted-model list with parameter estimates and route diagnostics.

frontier_fit_gls_psd *Experimental fitting route: frontier_fit_gls_psd*

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_gls_psd(model, data, optimizer = "nlopt-slsqp", control = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, missing = c("listwise", "error"), psd_factor = c("triangular", "square"))
```

Arguments

psd_factor	Covariance factor: triangular (default) or square.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.
start_eigen_floor	Eigenvalue floor used to construct a feasible covariance start.
feasibility_tol	Numerical tolerance for covariance feasibility.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.

Value

A fitted-model list with parameter estimates and route diagnostics.

```
frontier_fit_mixed_ordinal_psd
```

Experimental fitting route: frontier_fit_mixed_ordinal_psd

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_mixed_ordinal_psd(model, data, estimator = c("DWLS", "WLS", "ULS"), optimizer = "nlopt-slsqp", control = NULL, bounds = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, psd_factor = c("triangular", "square"))
```

Arguments

<code>psd_factor</code>	Covariance factor: triangular (default) or square.
<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>estimator</code>	Estimation objective. Supported objectives depend on the data kind and fitting route.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>bounds</code>	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
<code>start_eigen_floor</code>	Eigenvalue floor used to construct a feasible covariance start.
<code>feasibility_tol</code>	Numerical tolerance for covariance feasibility.

Value

A fitted-model list with parameter estimates and route diagnostics.

`frontier_fit_ml_multiinfo`

Experimental fitting route: `frontier_fit_ml_multiinfo`

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_ml_multiinfo(model, data, eta = 1.25, weight = NULL, optimizer = "port", control =
  bounds = NULL, missing = c("listwise", "error"), target = c("joint", "determinacy"))
```

Arguments

<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>eta</code>	Compatibility barrier parameter; used when weight is omitted.

weight	Weight selector or explicit fitting weight as required by the route.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.
bounds	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
target	Barrier target: joint covariance or factor-score determinacy.

Value

A fitted-model list with parameter estimates and route diagnostics.

frontier_fit_ml_psd *Experimental fitting route: frontier_fit_ml_psd*

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_ml_psd(model, data, optimizer = "nlopt-slsqp", control = NULL, start_eigen_floor =
  feasibility_tol = 1e-06, missing = c("listwise", "error"), preconditioning = c("diagonal",
    "none"), psd_factor = c("triangular", "square"))
```

Arguments

psd_factor	Covariance factor: triangular (default) or square.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.
start_eigen_floor	Eigenvalue floor used to construct a feasible covariance start.
feasibility_tol	Numerical tolerance for covariance feasibility.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
preconditioning	Optimizer-coordinate preconditioning: none or diagonal.

Value

A fitted-model list with parameter estimates and route diagnostics.

```
frontier_fit_ml_psd_fallback
```

Ordinary ML with Explicit PSD Recovery

Description

Fits ordinary complete-data normal-theory ML first, accepting it only when both the common fit verdict and covariance admissibility pass. Otherwise tries PSD-constrained ML once. The selection policy is implemented in C++.

Usage

```
frontier_fit_ml_psd_fallback(model, data, ordinary_optimizer = "nlopt-lbfgs", psd_optimizer = "nlopt-slsqp",
  psd_control = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, missing = c("listwise", "fiml",
    "error"), preconditioning = c("diagonal", "none"), psd_factor = c("triangular", "square"))
```

Arguments

psd_factor	Covariance factor: triangular (default) or square.
model	A model specification or model syntax string.
data	Sample statistics or a raw data frame.
ordinary_optimizer	Ordinary ML optimizer; defaults to NLOpt L-BFGS.
psd_optimizer	PSD optimizer; defaults to NLOpt SLSQP.
ordinary_control, psd_control	Separate optimizer control lists. Select the initial start policy with <code>ordinary_control\$start</code> ; <code>psd_control\$start</code> is rejected because the fallback determines its start.
start_eigen_floor	Positive floor used to initialize PSD factors, not a lower bound on the fitted covariance eigenvalues.
feasibility_tol	PSD covariance-link feasibility tolerance.
missing	Raw-data missing-value policy. This is not a FIML route.
preconditioning	PSD diagonal information scaling, or none.

Details

A warm start transfers ordinary parameter estimates only, not optimizer memory. The ordinary return must have finite parameters and objective, a positive-definite implied covariance, and satisfied model equalities to be used as a start. It need not pass accuracy or primitive-covariance admissibility. The PSD initializer projects covariance-factor starts; link equalities may initially be violated and are enforced by the solver. After an ordinary error or unusable return, PSD uses the original start.

Both stages must pass their common numerical verdict and covariance admissibility before their result can be selected. This does not certify global optimality or add boundary-aware inference. Ordinary ML defaults are unchanged. Parsing, data preparation, and initial start-construction errors occur before the fitting policy and raise R errors as usual.

Value

A list with `fit` (the accepted ordinary or PSD fit, otherwise NULL), `converged`, `fallback_used`, `fallback_reason`, and `warm_start_used`. The reason is "none", "ordinary-error", "ordinary-rejected", or "ordinary-inadmissible"; a failed accuracy verdict takes precedence over inadmissibility when both fail.

`ordinary` and `psd` retain each attempt as a list with `fit` and `error`, one of which is NULL. An error records `kind`, `detail`, `iterations`, and `f_value` from the core error value. The entire `psd` entry is NULL when skipped. Rejected returned estimates remain accessible even when `fit` is NULL.

Examples

```
s <- list(S = list(matrix(c(1, .7, .7, .7, 1, .3, .7, .3, 1), 3)),
  nobs = 300L)
result <- frontier_fit_ml_psd_fallback("f =~ x1 + x2 + x3", s)
result$fallback_reason
result$warm_start_used
```

`frontier_fit_ml2s_psd`

Experimental fitting route: `frontier_fit_ml2s_psd`

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_ml2s_psd(model, data, optimizer = "nlopt-slsqp", control = NULL, h_step = 1e-04, s,
  stage1_regularization = NULL, stage2_weight = "nt", dls_a = 0.5, start_eigen_floor = 1e-06,
  feasibility_tol = 1e-06, psd_factor = c("triangular", "square"))
```

Arguments

<code>psd_factor</code>	Covariance factor: triangular (default) or square.
<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>h_step</code>	Finite-difference step for the two-stage derivative calculation.
<code>stage1</code>	Optional precomputed Stage-1 fit for ML2S.
<code>stage1_regularization</code>	Optional Stage-1 regularization specification for ML2S.
<code>stage2_weight</code>	Two-stage fitting weight selector, such as nt or adf.
<code>dls_a</code>	DLS mixing scalar in $[0, 1]$: zero is normal theory and one is ADF.
<code>start_eigen_floor</code>	Eigenvalue floor used to construct a feasible covariance start.
<code>feasibility_tol</code>	Numerical tolerance for covariance feasibility.

Value

A fitted-model list with parameter estimates and route diagnostics.

`frontier_fit_ordinal_psd`

Experimental fitting route: frontier_fit_ordinal_psd

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_ordinal_psd(model, data, estimator = c("DWLS", "WLS", "ULS"), optimizer = "nlopt-s",
  bounds = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, psd_factor = c("triangu",
    "square"))
```

Arguments

<code>psd_factor</code>	Covariance factor: triangular (default) or square.
<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>estimator</code>	Estimation objective. Supported objectives depend on the data kind and fitting route.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>bounds</code>	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
<code>start_eigen_floor</code>	Eigenvalue floor used to construct a feasible covariance start.
<code>feasibility_tol</code>	Numerical tolerance for covariance feasibility.

Value

A fitted-model list with parameter estimates and route diagnostics.

`frontier_fit_sphere` *Experimental fitting route: `frontier_fit_sphere`*

Description

Fits in sphere coordinates with an explicit metric and pinning penalty, then optionally polishes in ordinary parameter coordinates. Inference remains a separate operation.

Usage

```
frontier_fit_sphere(model, data, estimator = "ML", groups = NULL, ..., cluster = NULL, ordered
  parameterization = "delta", psd = FALSE, missing = c("listwise", "error"), W = NULL, optim
  control = NULL, bounds = NULL, metric = c("unit_free", "raw"), pin_weight = 1, pole_tol =
  polish = TRUE, start = c("canonical", "user"), start_eigen_floor = 1e-06, feasibility_tol =
  preconditioning = c("none", "diagonal"), psd_factor = c("triangular", "square"))
```

Arguments

<code>psd_factor</code>	Covariance factor: triangular (default) or square.
<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

<code>estimator</code>	Estimation objective. Supported objectives depend on the data kind and fitting route.
<code>groups</code>	Grouping-variable name used when constructing a model from syntax.
<code>...</code>	Additional options forwarded to model construction or the underlying named operation.
<code>cluster</code>	Cluster-variable name or cluster identifiers for two-level fitting.
<code>ordered</code>	Names of ordinal observed variables.
<code>parameterization</code>	Ordinal model parameterization; currently delta.
<code>psd</code>	Whether to constrain model covariance blocks to be positive semidefinite.
<code>missing</code>	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
<code>W</code>	Fixed fitting-weight matrix, or a per-group list in model moment order.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>bounds</code>	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
<code>metric</code>	Sphere-coordinate metric: <code>unit_free</code> or <code>raw</code> .
<code>pin_weight</code>	Non-negative weight for the sphere pinning penalty.
<code>pole_tol</code>	Tolerance for detecting a sphere-coordinate pole.
<code>polish</code>	Whether to polish the sphere solution in ordinary parameter coordinates.
<code>start</code>	Start source: canonical construction or user hints.
<code>start_eigen_floor</code>	Eigenvalue floor used to construct a feasible covariance start.
<code>feasibility_tol</code>	Numerical tolerance for covariance feasibility.
<code>preconditioning</code>	Optimizer-coordinate preconditioning: <code>none</code> or <code>diagonal</code> .

Value

A fitted-model list with parameter estimates and route diagnostics.

`frontier_fit_uls_psd` *Experimental fitting route: `frontier_fit_uls_psd`*

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_uls_psd(model, data, optimizer = "nlopt-slsqp", control = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, missing = c("listwise", "error"), psd_factor = c("triangular", "square"))
```

Arguments

psd_factor	Covariance factor: triangular (default) or square.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
optimizer	Native optimizer name; NULL uses the route default.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.
start_eigen_floor	Eigenvalue floor used to construct a feasible covariance start.
feasibility_tol	Numerical tolerance for covariance feasibility.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.

Value

A fitted-model list with parameter estimates and route diagnostics.

frontier_fit_wls_psd *Experimental fitting route: frontier_fit_wls_psd*

Description

Fits the named objective through an explicit experimental covariance-domain route. PSD variants constrain covariance blocks; multiinfo variants apply the selected barrier target. Inference remains a separate operation.

Usage

```
frontier_fit_wls_psd(model, data, W, optimizer = "nlopt-slsqp", control = NULL, start_eigen_floor = 1e-06, feasibility_tol = 1e-06, missing = c("listwise", "error"), psd_factor = c("triangular", "square"))
```

Arguments

<code>psd_factor</code>	Covariance factor: triangular (default) or square.
<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>W</code>	Fixed fitting-weight matrix, or a per-group list in model moment order.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>start_eigen_floor</code>	Eigenvalue floor used to construct a feasible covariance start.
<code>feasibility_tol</code>	Numerical tolerance for covariance feasibility.
<code>missing</code>	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.

Value

A fitted-model list with parameter estimates and route diagnostics.

`frontier_newton_accuracy`

Local Newton accuracy diagnostic

Description

Computes the local Newton distance for complete-data ML in units of information-based standard errors. It is a quadratic approximation, not a global-optimality certificate. PSD fits can restrict the step to the active covariance face.

Usage

```
frontier_newton_accuracy(fit, budget = 0.01, psd = NULL)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>budget</code>	Positive allowed Newton distance in information-based standard-error units.
<code>psd</code>	Whether to constrain model covariance blocks to be positive semidefinite.

Value

A diagnostic list with status, distance, predicted gain and whether the requested budget passed.

frontier_reidentify *Reidentify a sphere fit*

Description

Projects a sphere-coordinate fitted model into the requested model identification.

Usage

```
frontier_reidentify(fit, model, ..., pole_tol = 1e-06)
```

Arguments

fit	A fitted magmaan model.
model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
...	Additional options forwarded to model construction or the underlying named operation.
pole_tol	Tolerance for detecting a sphere-coordinate pole.

Value

A fitted model in the requested identification with projection diagnostics.

global_score_flip_test
Global curved-SEM multiplier score test.

Description

Tests a fitted continuous ML, FIML, or normal-theory ML2S model against its local saturated mean/covariance alternative. ML/FIML use direct saturated likelihood scores. ML2S uses the Stage-2 saturated score for the observed gate and propagates Stage-1 EM case-wise moment influence through the fitted normal-theory metric for multiplier calibration. In each case **df** is the saturated moment dimension minus the tangent rank. This is the general goodness-of-fit counterpart to `score_flip_test()`, which instead releases specified affine restrictions.

Usage

```
global_score_flip_test(
  fit,
  data = NULL,
  n_flips = 999L,
  seed = 1,
  multiplier = c("rademacher", "mammen", "two-point", "gaussian", "centered-exponential"),
  two_point_skewness = 1,
  center_multiplier_scores = FALSE,
  multiplier_studentization = c("none", "weighted-meat"),
  sensitivity = c("expected", "observed", "observed-shrink-light",
    "observed-shrink-sqrt", "observed-h1"),
  metric = c("expected", "observed", "observed-h1")
)
```

Arguments

fit	A fitted continuous ML, FIML, or normal-theory ML2S <code>magmaan</code> model.
data	Raw fitting data for complete-data ML. FIML and ML2S use <code>fit\$raw_data</code> .
n_flips	Number of random multiplier transformations.
seed	Non-negative deterministic integer seed.
multiplier	Multiplier distribution. Non-Rademacher resampling requires <code>calibration = "effective"</code> .
two_point_skewness	Non-negative third moment for <code>multiplier = "two-point"</code> . The corresponding fourth moment is the minimum possible value, $1 + \text{two_point_skewness}^2$; zero gives a Rademacher law and one gives Mammen's law.
center_multiplier_scores	Whether to center effective-score contributions within information strata before applying multipliers.
multiplier_studentization	Optional "weighted-meat" bootstrap-t diagnostic using the per-draw meat $\sum(w_i^2 u_i u_i')$. "none" retains the expected-information metric.
sensitivity	Nuisance-projection bread. "expected" uses the fitted H0 Fisher information, "observed" uses the realized H0 Hessian, and "observed-shrink-light" and "observed-shrink-sqrt" shrink that Hessian toward expected Fisher information using a predeclared function of the nuisance-tangent dimension divided by sample size. Both weights vanish under fixed-dimensional asymptotics. Finally, "observed-h1" uses the realized saturated H1 Hessian. The H1 choice is null-equivalent to observed H0 sensitivity but evaluated at the stable saturated optimum. The observed choices are unavailable for ML2S.
metric	Score-quadratic and robust-spectrum bread. "expected" uses the stable pattern-conditional Fisher information. "observed" uses the realized H0 likelihood information and fails if it is not positive definite on the

tested complement. "observed-h1" uses the realized saturated likelihood information at the fitted H1 moments; under a pseudo-null it has the same population target as observed H0 information but is positive definite at the saturated optimum. The observed choices are experimental ML/FIML diagnostics and are not defined for ML2S.

Details

The default sensitivity is observed for FIML under MAR and expected for complete-data global goodness-of-fit under the correct-model null. Explicit sensitivity and metric choices remain available.

Value

A magmaan_global_score_flip_test list containing the effective multiplier result, asymptotic comparators, and saturated/tangent geometry diagnostics.

guttman_h	<i>Estimate the diagonal of the Guttman H matrix.</i>
-----------	---

Description

Estimate the diagonal of the Guttman H matrix.

Usage

```
guttman_h(  
  S,  
  blocks,  
  method = c("triad_ls", "extended_triad_ls", "triad_mean", "triad_pooled", "triad_wls",  
    "triad_wls_joint", "ar", "rs", "gmm_block", "gmm_full", "anchor_triad_ls", "ilm",  
    "anchor_ilm")  
)
```

Arguments

S	Observed covariance or correlation matrix.
blocks	One simple-structure factor/block label per observed variable.
method	Communality rule, named as [extended_]triad_[mean pooled ls wls wls_joint]: "triad_ls" (identity-weighted triad least squares), "extended_triad_ls" (same plus cross-block triads, leveraging factor correlations), "triad_pooled" (pooled ratio of triad sums), "triad_mean" (mean of triad ratios; the lavaan rule), "triad_wls" (efficiency-weighted triad LS, per block), or "triad_wls_joint" (one joint weight across blocks). Old names "ar", "rs", "gmm_block", "gmm_full", "anchor_triad_ls", "ilm", and "anchor_ilm" are still accepted as aliases.

Value

A list with correlation-scale communalities `h2`, covariance-scale diagonal `h_diag`, and `H`, equal to `S` with the diagonal replaced by `h_diag`.

<code>inference_rows</code>	<i>Casewise rows of a prepared quadratic</i>
-----------------------------	--

Description

Extracts casewise contributions in group-block order from a prepared inference quadratic.

Usage

```
inference_rows(quadratic)
```

Arguments

`quadratic` Prepared quadratic from `inference_quadratic`.

Value

A matrix of casewise contributions.

<code>lab_inference_primitives</code>	<i>Explicit geometry and sampling laws for lab primitives</i>
---------------------------------------	---

Description

Methods-development primitives exposed through `magmaan_core`. Their ingredients must describe the component's null and sampling law.

Usage

```
infer_build_u_factor(fit, bread, moments = "structured")
infer_build_u_factor_parts(partable, sample_stats, theta, bread,
  moments = "structured")
infer_build_u_factor_pairwise(fit, X, bread, mask = NULL)
infer_build_u_factor_fit(fit, bread, moments = "structured")
infer_continuous_ls_robust(fit, raw_data = NULL, weight = NULL,
  bread = "observed", gamma = NULL, fixed_weight = FALSE)
infer_continuous_ls_profile_lrt(fit_H1, fit_H0, X_per_group = NULL,
  weight = NULL, eig_tol = 1e-10, gamma = NULL)
infer_ml_profile_lrt(fit_H1, fit_H0, X_per_group = NULL,
  eig_tol = 1e-10, gamma = NULL)
measures_reliability_cov(S, gamma = NULL, n = 0L, raw_data = NULL)
measures_reliability_omega_multidim(S, block, target = "total",
  weights = NULL, gamma = NULL, n = 0L, raw_data = NULL)
```

Arguments

fit	Fitted model.
fit_H1, fit_H0	Less and more restricted fits for the profile LRT.
X_per_group	Raw observations in fitted group order, used when gamma is NULL. Omit when supplying Gamma.
eig_tol	Relative tolerance for profile-law eigenvalues.
bread	Required geometry for U factors: expected for complete-data global goodness-of-fit under the correct-model null; observed requests the explicit Hessian convention. U factors alone do not certify pseudo-true covariance or nested sampling laws.
moments	For U factors, "structured" or "unstructured" selects the fitted/sample normal-theory weight, not centering. These correct-null GOF conventions retain their defaults and reject "likelihood" . Lab ML SE/score meats instead default to "auto" : exact uncentered likelihood projections for empirical covariance, structured for model-implied comparators. Exact likelihood projections include fitted-mean shifts and group score means; centered Zc and caller Gamma use the same affine transformation.
partable, sample_stats, theta	Model, observed moments and fitted parameters.
X, mask	Raw observations and optional observation mask. Pass mask by name in the pairwise builder.
raw_data	Raw fitting observations; a numeric matrix for reliability.
weight	Fitting weight, optionally retained on the fit.
gamma	For LS robust covariance, NULL retains empirical covariance from raw_data ; "empirical" and explicit "normal" sources remain supported. A supplied symmetric PSD matrix (one group) or list of per-group matrices replaces the meat and requires fixed_weight = TRUE . ML/LS profile LRTs also accept these matrices; NULL builds empirical Gamma from X_per_group . Blocks use means first when fitted, then lower-triangle covariances by columns including diagonals, in fitted observed-variable order. Supply unscaled per-group NACOV (N times sampling covariance); group weights are applied by C++. Dimensions, finiteness, symmetry and PSD are validated. For reliability, Gamma is an explicit asymptotic covariance of vech(S) ; NULL derives empirical Gamma from raw_data and errors if those observations are absent.
fixed_weight	Explicitly request the caller-fixed-weight LS law. For data-estimated nt/adf/dwls/dls recipes, use infer_casewise_influence_ij_fit() for parameter covariance. This primitive errors unless fixed_weight = TRUE requests its comparator. Grouped sample-centered moment meats describe fixed group allocation; random composition with nonzero group estimating-equation means requires additional allocation influence.
S	Observed covariance matrix in the same metric as Gamma.
n	Sample size; inferred from reliability raw_data if zero.
block	Integer item-to-factor allocation.

target Total or hierarchical omega.
weights Optional item weights.

Details

Scalar `frontier_profile_lrt_*` and interval primitives default to the misspecification-scaled reference (`MisspecScaled`) when `reference = NULL`, `robust = FALSE`. Supply raw data where required. The explicit `reference = "ordinary"` and `reference = "robust_scaled"` comparators remain available. Explicit `robust = TRUE` with `NULL` reference retains `RobustScaled`. Named Satorra, lavaan bundle and FMG/semTests replication APIs retain their documented compatibility conventions. SAM and case-change SE/type options standardize diagnostics and do not certify a generic inferential covariance.

<code>lav_residuals</code>	<i>Standardized residual diagnostics</i>
----------------------------	--

Description

Returns standardized residual matrices, residual SE and z statistics, SRMR and block summaries. Estimated-weight inference for continuous least squares requires raw observations.

Usage

```
lav_residuals(fit, estimated_weight = TRUE, data = NULL)
```

Arguments

fit A fitted magmaan model.
estimated_weight Whether to include estimated-weight influence in continuous least-squares residual inference; requires raw data.
data Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Value

A list of residual matrices and block-level diagnostic summaries.

magmaan_core	<i>Composable native primitives</i>
--------------	-------------------------------------

Description

A named list of low-level native operations, organized by domain. Use the exported preparation, fitting and inference wrappers when they cover the required operation.

Usage

```
magmaan_core
```

Value

A list of callable native primitives.

magmaan_fcsem	<i>Factor-component SEM: magmaan_fcsem</i>
---------------	--

Description

Constructs, fits or summarizes the native factor-component SEM representation. These objects use their own fitting route.

Usage

```
magmaan_fcsem(model, data, ..., missing = c("listwise", "error"), control = NULL)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
...	Additional options forwarded to model construction or the underlying named operation.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
control	Named optimizer control list, including iteration limits, tolerances and optional starts.

Value

A factor-component model specification, fitted model or named post-fit summary, according to the operation.

<code>mahalanobis_rerun</code>	<i>Mahalanobis distance of each case on the observed variables</i>
--------------------------------	--

Description

Model-free leverage measure: `stats::mahalanobis(X, colMeans(X), stats::cov(X))` (N-1 covariance), needing no refit. Mirror of `semfindr::mahalanobis_rerun()`. For multiple-group fits the distance is computed within each group (its own mean/cov); pass the original data frame to place the per-group distances back at the original rows (semfindr order).

Usage

```
mahalanobis_rerun(fit, data = NULL)
```

Arguments

<code>fit</code>	A <code>magmaan_fit</code> or a <code>magmaan_case_rerun</code> .
<code>data</code>	Optional original data frame (multiple-group fits, to recover the original row order). If omitted, multiple-group distances are returned in group-block order.

Value

A one-column matrix of Mahalanobis distances, rownames = case ids.

<code>mi_screen</code>	<i>Screen modification diagnostics by significance and effect</i>
------------------------	---

Description

Adds multiplicity-adjusted p-values and a verdict based on significance and absolute effect size. Missing inputs produce missing verdicts.

Usage

```
mi_screen(x, p = NULL, effect = NULL, alpha = 0.05, effect_min = 0.1, adjust = "BH")
```

Arguments

<code>x</code>	Input data frame; for <code>mi_screen</code> , a modification-index or score-test results table.
<code>p</code>	Name of the p-value column; NULL selects a recognized robust column.
<code>effect</code>	Name of the effect-size column; NULL selects a recognized robust column.
<code>alpha</code>	Significance threshold after multiplicity adjustment.
<code>effect_min</code>	Minimum absolute effect size considered substantively large.
<code>adjust</code>	Multiplicity method passed to <code>stats::p.adjust</code> .

Value

The input table with adjusted p-values and verdict (free, trivial, underpowered or ok).

<code>model_spec</code>	<i>Construct a lavaan-syntax model specification</i>
-------------------------	--

Description

Constructs the model structure, names and start hints used by the lab fitting interfaces.

Usage

```
model_spec(syntax, ..., auto_var = TRUE, auto_cov_lv_x = TRUE, auto_cov_y = FALSE,
  orthogonal = FALSE, auto_fix_first = TRUE, marker = NULL, auto_fix_single = TRUE, std_lv =
  effect_coding = FALSE, fixed_x = TRUE, meanstructure = "default", model_type = c("sem",
    "growth"), ordered = NULL, parameterization = "delta", group = NULL, group_labels = NU
  group_equal = NULL, group_partial = NULL)
```

Arguments

<code>syntax</code>	A single model-syntax string.
<code>...</code>	Additional options forwarded to model construction or the underlying named operation.
<code>auto_var</code>	Automatically add variances for model variables.
<code>auto_cov_lv_x</code>	Automatically add covariances among exogenous latent variables.
<code>auto_cov_y</code>	Automatically add covariances among dependent variables.
<code>orthogonal</code>	Fix covariances among latent variables to zero.
<code>auto_fix_first</code>	Use a marker loading to identify each latent variable.
<code>marker</code>	Named character vector mapping latent names to marker indicators. With <code>auto_fix_first</code> , select the first matching loading or fall back to the first loading; preserve row order and apply user modifiers normally.
<code>auto_fix_single</code>	Fix the residual variance of a single indicator to zero.
<code>std_lv</code>	Identify latent scales by unit latent variances.
<code>effect_coding</code>	Effect-code loadings or intercepts when supported.
<code>fixed_x</code>	Treat observed exogenous variables as fixed.
<code>meanstructure</code>	Whether to include intercepts and means; <code>model_spec</code> also accepts the default convention.
<code>model_type</code>	Model construction mode: sem or growth.
<code>ordered</code>	Names of ordinal observed variables.
<code>parameterization</code>	Ordinal model parameterization; currently delta.

<code>group</code>	Grouping-variable name or group identifiers, as appropriate for the data.
<code>group_labels</code>	Explicit group labels in block order.
<code>group_equal</code>	Parameter classes constrained equal across groups.
<code>group_partial</code>	Named exceptions to the group equality constraints.

Value

A model specification containing the projected parameter table and construction metadata.

<code>modification_indices</code>	<i>Post-fit parameter diagnostics: modification_indices</i>
-----------------------------------	---

Description

Computes explicit release or restriction diagnostics. LR-type variants use the supplied observations for robust spectral inference. Unsupported candidate regimes are reported explicitly.

Usage

```
modification_indices(fit, data = NULL, ..., candidates = "all")
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>...</code>	Additional options forwarded to model construction or the underlying named operation.
<code>candidates</code>	Candidate parameter rows to release or test: all or the named subset supported by the operation.

Value

A table of candidate parameter diagnostics.

modification_indices_lrt

Post-fit parameter diagnostics: modification_indices_lrt

Description

Computes explicit release or restriction diagnostics. LR-type variants use the supplied observations for robust spectral inference. Unsupported candidate regimes are reported explicitly.

Usage

```
modification_indices_lrt(fit, data, candidates = c("all", "loadings", "covariances"), robust =
  weight = NULL)
```

Arguments

fit	A fitted magmaan model.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
candidates	Candidate parameter rows to release or test: all or the named subset supported by the operation.
robust	Whether to compute robust LR-type diagnostics.
weight	Weight selector or explicit fitting weight as required by the route.

Value

A table of candidate parameter diagnostics.

modification_indices_robust

Robust modification indices and equality-release score tests

Description

Compute one-parameter modification indices or one-at-a-time equality-release tests, with a direction-specific sandwich correction.

Usage

```
modification_indices_robust(fit, data = NULL, weight = NULL,
  bread = "observed", moments = "auto", cov = "empirical",
  candidates = "all", include_loadings = TRUE, include_covariances = TRUE,
  information = "expected", estimated_weight = NULL, gamma = NULL)
score_tests_robust(fit, data = NULL, weight = NULL,
  bread = "observed", moments = "auto", cov = "empirical",
  estimated_weight = TRUE, gamma = NULL)
```

Arguments

fit	A fitted magmaanlab model.
data	Fitting observations as a data frame, matrix, or grouped raw-data blocks. FIML uses retained observations when this is omitted; an explicit FIML raw-data object may also carry a missingness mask. Two-stage (ML2S) fits use their retained Stage-1 data and reject data .
weight	Fitting weight for continuous WLS-computed fits. Fits record their weight in fit\$W , so this is needed only for fits without that record; otherwise it must equal the recorded weight or a positive multiple of it. Unsupported for FIML and ML2S.
bread, information	Sandwich sensitivity and score-information choices. For ML/FIML, bread selects nuisance sensitivity; the metric and EPCs use expected information, with either information selector accepted. FIML requires bread="observed" . Bread defaults to observed for all estimators.
moments, cov	Continuous moment and covariance conventions. moments="auto" selects exact likelihood-score meat for empirical complete-data ML, including fitted-mean shifts and joint group-score means. Explicit "structured" and "unstructured" select fitted/sample normal-theory weights on centered moment influence. Continuous LS retains its fitting-weight moment law (fixed group allocation for centered group meats). FIML auto selects moments="structured" , cov="empirical" combination, which selects observed-pattern casewise score covariance.
candidates	"fixed" for existing fixed parameter rows or "all" to include absent cross-loadings, covariances and regressions among variables already participating in structural equations, including reverse paths. Unidentified regression directions retain unavailable rows.
include_loadings, include_covariances	Whether to enumerate absent loading and covariance candidates.
gamma	Optional symmetric positive semidefinite caller NACOV: a matrix for one group or a list of matrices in fitted group order. Continuous moments are means first when fitted, then lower-triangle covariances by columns, including the diagonal, in the model's observed-variable order. All-ordinal moments are thresholds then lower-triangle polychorics by columns, excluding the diagonal. Mixed moments are thresholds, negative continuous means, continuous variances, then lower-triangle associations. Each block is the unscaled per-group NACOV (N times the sampling covariance); ML's group n/N weights are applied by the adapter. Dimensions, finite entries, symmetry and PSD are validated (relative tolerance 1e-10 for symmetry and PSD). Supplying this replaces the covariance meat and preserves the fitting weight. It requires estimated_weight = FALSE explicitly and cov = "empirical" ; FIML and ML2S refuse caller Gamma. NULL preserves the usual covariance source. A model-implied Gamma is an explicit choice: use cov = "model_implied" where supported, or supply a normal-theory matrix constructed explicitly for the chosen moments.

estimated_weight

For modification indices, NULL selects FALSE for ML/FIML and TRUE for estimators with a second-stage fitting weight. Include supported estimated-weight influence terms, following the weight recipe the fit records (NT, ADF, DWLS, or DLS with its mixing weight; ULS has none). Supplied weights have no recipe and ordinal NT and DLS weights have no derived influence, so both fail with an **UnsupportedInference** error. FIML has no second-stage weight and rejects this option. Supplied Gamma cannot carry casewise weight influence and also rejects this option.

Details

ML/FIML use expected information as the metric and bread, observed nuisance sensitivity when **bread="observed"**, and empirical score meat (FIML uses observed-pattern scores). EPCs use the expected-information projection. Negative observed release curvature does not prevent a robust statistic. It needs no saturated-model EM computation for these tests. Explicit data rebuild their missingness pack; omitted data reuse the retained pack when available. Alternative FIML moment or covariance recipes and supplied second-stage weights are rejected.

Continuous ULS/GLS/WLS support empirical covariance from fitting observations, or model-implied normal-theory covariance from structured or unstructured moments. WLS-computed fits (WLS, DWLS, DLS, supplied W) use the recorded fitting weight for either source; the correction reduces to the ordinary statistic only when that weight is the inverse of the chosen covariance. Browne-unbiased covariance is not implemented for continuous LS. Estimated-weight mode requires raw observations and empirical covariance; other covariance choices are rejected.

Two-stage (ML2S) fits report **mi** as the naive Stage-2 statistic on the Stage-1 EM moments (attribute **mi_type = "naive_stage2"**) and **mi.scaled** with the Stage-1 moment covariance as meat. They use the recorded Stage-2 weight, expected information and the default **bread**, **moments** and **cov**; a DWLS, ADF or DLS Stage-2 weight adds its data influence with **estimated_weight = TRUE**. The naive NT row uses the structured information; lavaan's two-stage **modindices()** uses the unstructured one and differs.

Identification-only releases are excluded. Ordinal and mixed-ordinal methods use their retained categorical moment covariance and weight conventions; the all-ordinal **bread** selects observed or expected nuisance sensitivity, and mixed methods now honor the same selection. Complete mixed estimated-weight MI/releases use exact empirical first-stage rows and mixed Gamma influence. Expected mixed projection and caller-Gamma comparators retain NACOV. All-ordinal **moments** and **cov** choices use the retained NACOV.

Generic **modification_indices()** and **score_tests()** compose empirical robust inference. To request the historical unscaled comparator explicitly, use **cov = "model_implied"**, **estimated_weight = FALSE**, and the desired bread/information. Continuous ULS/GLS/DWLS/WLS/DLS and all-ordinal ULS/DWLS/WLS and complete mixed DWLS/WLS support observed-Hessian nuisance sensitivity with estimated-weight influence. Their quadratic metric stays expected and their reference law uses the matching projected meat. Selecting expected bread or fixed weights requests a comparator. At exact fit the observed and expected projections coincide and the weight-influence correction vanishes. Observed-sensitivity ML2S score laws remain unavailable.

Value

Tables carry **reason** and **detail**; identified numerical failures retain their candidate rows with NA values. A data frame with candidate names and groups, unscaled quadratic **mi**, EPCs, **mi.scaled**, **scaling.factor**, and the robust **pvalue**.

mplus_data	<i>Read data described by a Mplus model</i>
------------	---

Description

Reads numeric individual or summary ASCII data using the C++ data plan. Relative paths are tried in the working directory, then beside the input file. Free format accepts blanks, tabs and commas, wrapped observations, and discards extra fields at the end of a completed observation's record. Fixed FORMAT supports Fw.d, Fw or w.d, X skips, Th positions, record breaks and repeated groups. Implied decimals apply only without an explicit point. Missing flags compare after that scaling. A terminal DOS Ctrl-Z, optionally followed only by whitespace, denotes EOF. Embedded EOF markers and other control bytes are rejected; remove the marker or the trailing records.

Usage

```
mplus_data(model, file = NULL)
```

Arguments

model	A magmaan_mplus_model_spec from mplus_model() .
file	Optional path override, one path per separate-file group.

Details

Separate-file groups use a reserved **.mplus_group** column containing FILE labels in declaration order. NGROUPS summary groups are named g1, g2, ... GROUPING drops unlisted codes and reports counts. NOBSEVATIONS limits individual observations per file. LISTWISE and analysis-sample deletion rules are only reported: use `fit_model(..., missing = "listwise")` to remove incomplete rows or `missing = "fiml"` for FIML; explicitly filter missing covariates to reproduce Mplus's conditional analysis sample.

Summary matrices use divisor N-1 on input. As in Mplus ML and lavaan's default `sample.cov.rescale`, they are multiplied by (N-1)/N for `fit_model()`. Correlations are multiplied by supplied SDs; without SDs the input has unit variances. The report retains input covariances and conversion factors.

Accepted summary TYPE values are COVARIANCE, CORRELATION, FULLCOV and FULLCORR, optionally with MEANS (and STDEVIATIONS for correlations). Supply NOBSEVATIONS for every summary group; summary input uses free format. MISSING accepts one global ., * or fixed-format BLANK symbol, or numeric flags per NAMES list/ALL. Fixed blanks without BLANK missing read as zero. Empty comma fields are errors; numeric records are bounded to 10000 characters. File and matrix shape errors

identify the data rule. Perform DEFINE/qualified DATA transformations and USEOBSERVATIONS or SUBPOPULATION selection in R before reading/fitting. Multiple imputation, Monte Carlo datasets and two-level weight matrices are outside this reader; supply one supported individual or summary dataset per declared group.

Value

Individual data as a `data.frame`, or summary moments as a list with `S`, mean and nobs accepted by `fit_model()`. Attribute `mplus_data_report` records paths, counts, dropped codes and unapplied sample rules.

<code>mplus_model</code>	<i>Construct a model from a Mplus input file</i>
--------------------------	--

Description

Imports single- and multiple-group linear SEM: BY, ON, WITH, PON, PWITH, means, variances, starts, fixes, labels and equality numbers, NOCOVARIANCES, and NOMEANSTRUCTURE with INFORMATION = EXPECTED. Names resolve without regard to case and retain the spelling in NAMES; factors retain their first BY spelling. Estimator and execution settings are reported, not imported.

Usage

```
mplus_model(input = NULL, file = NULL)
```

Arguments

<code>input</code>	One string containing the whole input text.
<code>file</code>	Path to an input file. Supply exactly one of input or file.

Details

CATEGORICAL lists binary and ordinal outcomes (two through ten categories). Thresholds use `[u$k]`, DELTA scales use `{u}`, and THETA residuals use variance statements. Categories and thresholds are completed from data; all groups must contain the same categories. All-ordinal models fit with `estimator = "DWLS"`; mixed, conditional and ML categorical routes report their unsupported route explicitly. Retained-estimate lavaan WLSMV reporting is available through `convention_inference()`.

Fixed-time polynomial and piecewise growth (`i s | y1@0 y2@1 ...`) imports outcome intercept, growth mean, threshold and scale/residual defaults. Growth accepts one through four factors; free time scores are accepted when enough scores remain fixed to identify the polynomial loadings. MODEL CONSTRAINT imports NEW (starts default to 0.5), explicit and implicit equalities, derived quantities, and nested DO loops. Functions include EXP, LOG, SQRT, PHI and LOG10; `**` denotes power. Continuous ML, LS and FIML use the existing equality backends. All-ordinal nonlinear equalities use constrained LS fitting and expected-information lavaan reporting. Native policy and observed/IJ inference require

unimplemented Lagrangian curvature. Inequalities are deliberately refused: active-bound inference requires boundary asymptotics; for variance positivity drop the constraint and use `covariance = "psd"` or `"barrier"` in `fit_model()`. MODEL INDIRECT imports total IND, specific IND and VIA products, including paths through factors. Derived names are `ind_g<group>_<outcome>_ind_<names>` or `ind_g<group>_<outcome>_via_<mediator>_<predictor>`; specific names retain the written mediator order. Defined estimates and delta-method SEs are available through `compute_defined()`. A free NEW coordinate appears as a **new** partable row, with no observed variable or moment cell; rebuilding always uses the original Mplus source. INFORMATION defaults, MODEL TEST, LOOP and PLOT are reported in notes.

NAMES accepts bounded numeric- or letter-suffix ranges; MODEL and USEVARIABLES ranges follow schema order. Latent ranges follow factor-definition order. Labels apply on their physical line; subsequent mentions override earlier parameter specifications. PON and PWITH require equally sized lists. Model/option content must fit within 90 columns; wrap longer statements. TITLE and comments may be longer. Line and block comments are accepted, but a block opened after code must close on its opening line. TITLE, OUTPUT, SAVEDATA and PLOT are reported.

DEFINE and DATA transformations must be performed in R; USEOBSERVATIONS and SUBPOPULATION must be applied as case selection in R before fitting. Counts, nominal/censored/survival outcomes, survey weights, mixtures, multilevel models, Bayes and full-information categorical links are outside the linear SEM family: retain those analyses in Mplus. ESEM rotation is replaced only by explicitly specified ordinary BY factors. Bare `@` uses data-dependent Mplus starts; write an explicit `@value` instead. Remove all observed-independent means/variances/WITH mentions to preserve fixed-x conditioning, or bring every observed independent variable into the joint random-X model (e.g. `x1 x2;`). Complete mentions free X means, variances and default covariances; partial mentions are rejected with the exact variance statement that completes the joint model. Recode data-dependent GROUPING forms to explicit integer code/label pairs in R. Categorical summary inputs require individual observations for magmaan's moment preparation.

DEFINE, mixtures, multilevel models, ESEM, unsupported name ranges and mixed conditioning on observed covariates are rejected with rule IDs and an explanation of what to write instead. Data descriptions are parsed into `$mplus_data_plan`; `mplus_data()` reads them. This constructor does not read the data file or run Mplus. GROUPING imports explicit integer code = label pairs, ordered by numeric code. Group MODEL sections override the overall model; repeated sections apply cumulatively. CONFIGURAL, METRIC and SCALAR accept one setting at a time. Categorical shortcuts accept CONFIGURAL and SCALAR; METRIC is rejected for binary and ordinal outcomes, following Mplus 9.1. Group-only changes to variable roles are rejected; put the relation in the overall model and fix it in the other groups instead. Variance identification in a shortcut uses free first loadings and explicit factor variances at one. `mplus_data()` drops unlisted GROUPING codes with counts. Separate FILE groups retain source labels, use `.mplus_group`, and follow FILE order. Summary NGROUPS labels are `g1, g2, ...`. Summary data without MEANS have no mean structure and reject explicit means/intercepts.

Value

A `magmaan_model_spec` with original `mplus_source`, a `data.frame` of `mplus_notes` (class, rule, line, col, message), and a lavaan syntax projection. Grouped specs also carry `group_var`,

ordered integer-code group_labels and mplus_groups (a data.frame with label and code columns). Rebuilding uses mplus_source; construction options cannot be overridden.

`mplus_wlsmv_invariance`

Compose ordinal invariance fits

Description

Constructs and fits the requested ordinal invariance sequence using the Mplus-style WLSMV conventions.

Usage

```
mplus_wlsmv_invariance(model, data, ordered, group, steps = c("configural", "metric", "scalar"),
  estimator = c("WLSMV", "DWLS", "WLS"), missing = c("listwise", "pairwise", "error"),
  group_labels = NULL, optimizer = "nlopt-lbfgs", control = NULL, bounds = NULL,
  pd_gamma = c("overlap", "nominal"), validate = TRUE)
```

Arguments

<code>model</code>	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
<code>data</code>	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
<code>ordered</code>	Names of ordinal observed variables.
<code>group</code>	Grouping-variable name or group identifiers, as appropriate for the data.
<code>steps</code>	Invariance steps to construct: configural, metric or scalar.
<code>estimator</code>	Estimation objective. Supported objectives depend on the data kind and fitting route.
<code>missing</code>	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
<code>group_labels</code>	Explicit group labels in block order.
<code>optimizer</code>	Native optimizer name; NULL uses the route default.
<code>control</code>	Named optimizer control list, including iteration limits, tolerances and optional starts.
<code>bounds</code>	Lower and upper parameter-bound vectors or a named bounds preset; NULL leaves the route default.
<code>pd_gamma</code>	Pairwise fourth-moment covariance convention: overlap or nominal.
<code>validate</code>	Validate the constructed invariance models.

Value

An invariance result with the constructed models, fits and comparisons.

<code>nested_score_test</code>	<i>Nuisance-effective nested score test without random sign calibration.</i>
--------------------------------	--

Description

Computes the same nuisance-orthogonalized observed score and robust restriction spectrum used by `score_flip_test()`, but skips every random sign draw and the flip-specific covariance calculations. The recommended `p_value` is the pEBA4 mixture approximation. Scalar SB, exact-mixture, and direct sandwich comparators remain available for diagnostics.

Usage

```
nested_score_test(
  fit_H1,
  fit_H0,
  data = NULL,
  sensitivity = c("observed", "expected")
)
```

Arguments

<code>fit_H1</code>	Less-restricted complete-data ML or direct-FIML fit, or a <code>magmaan_model_spec</code> for the less-restricted model. Supplying the model avoids fitting H1 because a score test uses only H1's tangent and H0's estimates.
<code>fit_H0</code>	More-restricted nested fit with paths dropped, fixed or constrained, using the same estimator and observations. Interior moment reparameterizations are evaluated at a common null point.
<code>data</code>	Raw fitting data for complete-data ML. Direct FIML uses the observed-data sample stored on both fits, so this argument must be omitted.
<code>sensitivity</code>	Nuisance sensitivity (bread). "expected" preserves the original pattern-conditional normal-theory construction; "observed" uses the realized likelihood Hessian for the nuisance projection and supplies the pseudo-true/MAR correction when information equality need not hold. The score quadratic retains the expected metric.

Value

A list of class `magmaan_nested_score_test` containing the effective score statistic, restriction eigenvalues, pEBA4/SB/exact-mixture/sandwich p-values, conditioning diagnostics, and setup/asymptotic timings.

```
noniterative_cfa_constrained
```

Linearly-constrained non-iterative CFA fit (measurement invariance).

Description

Minimum-distance projection of the configural (per-group) fit onto the linear equality constraints $A \theta = b$ carried by the partable's `group.equal` family (metric / tau-equivalence / strict invariance, fixed values, ...). The Omega-metric projection is closed-form: $\theta_{\text{tilde}} = \theta_{\text{hat}} - \Omega A'(A \Omega A')^{-1} (A \theta_{\text{hat}} - b)$, and the constraint statistic W is an exact chi-square on k (Wald = minimum-distance duality). Returns the projected estimate (`theta_tilde`), its rank-deficient covariance (`vcov_constrained`, `se_constrained`), and the constraint test (W , k , p_{wald}).

Usage

```
noniterative_cfa_constrained(
  fit,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>discrepancy</code>	Accepted for compatibility; ignored because this wrapper only needs the delta-method covariance.
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If NULL, infer the estimator map from <code>fit</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

```
noniterative_cfa_difference_test
```

Difference / pseudo-LRT test between two nested non-iterative CFA fits.

Description

$T_d = T(H_0) - T(H_1)$ with a weighted-chi-square reference (H_1 -anchored). Secondary to the Wald test: T_d can be negative for a non-minimizing estimator (flagged in `warnings`).

Usage

```
noniterative_cfa_difference_test(
  fit0,
  fit1,
  df_d,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data0 = NULL,
  data1 = NULL,
  estimator = NULL
)
```

Arguments

<code>fit0</code>	The more-restricted (H_0) fit.
<code>fit1</code>	The less-restricted (H_1) fit.
<code>df_d</code>	Degrees of freedom of the restriction ($q_{H1} - q_{H0}$).
<code>discrepancy</code>	"uls" (unweighted least squares) or "ntml" (model-implied normal-theory / reweighted least squares).
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data0, data1</code>	Raw data for each fit when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see fit_noniterative_cfa() . If NULL, infer the estimator map from <code>fit</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

noniterative_cfa_grouped_inference

Grouped (multi-group / mean-structure) residual-based inference.

Description

Fits every group's Guttman block independently and stacks them: a block-diagonal delta-method covariance (`vcov`, `se`), a joint goodness-of-fit `T` on `df` degrees of freedom with the union weighted-chi-square reference, and `block_of_param` (0-based group index per free parameter). Reduces to `noniterative_cfa_inference()` for a single group with no mean structure. Required for mean-structure models (the single-group path omits the intercept block).

Usage

```
noniterative_cfa_grouped_inference(
  fit,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>discrepancy</code>	"uls" (unweighted least squares) or "ntml" (model-implied normal-theory / reweighted least squares).
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If <code>NULL</code> , infer the estimator map from <code>fit</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

noniterative_cfa_inference
Residual-based inference for a non-iterative CFA fit.

Description

Returns delta-method standard errors (**se**, **vcov**), the goodness-of-fit statistic **T** on **df** degrees of freedom, its weighted-chi-square p-value family (**p_scaled** Satorra-Bentler, **p_meanvar**, **p_scaled_shifted**, **p_mixture**), the reference eigenvalues, and the Satorra-Bentler **scale_c**. For **discrepancy** = "ntml", **rls_check** is the reweighted-least-squares cross-check of **T**.

Usage

```
noniterative_cfa_inference(
  fit,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL
)
```

Arguments

fit	A fit from <code>fit_noniterative_cfa()</code> .
discrepancy	"uls" (unweighted least squares) or "ntml" (model-implied normal-theory / reweighted least squares).
gamma	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires data).
data	Raw data matrix (rows = observations), required when gamma = "empirical".
estimator	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If NULL , infer the estimator map from fit .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

noniterative_cfa_modification_indices

Residual-based modification-index diagnostics for a non-iterative CFA fit.

Description

Closed-form Guttman fits are not stationary minimizers of the ML/ULS discrepancy, so the usual score-test modification index is not defined. This helper reports residual-space analogues for fixed-zero and absent one-parameter candidates: raw residual scores, tangent-residualized residual scores, and local one-step discrepancy-drop/EPC diagnostics.

Usage

```
noniterative_cfa_modification_indices(
  fit,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL,
  candidates = c("all", "fixed"),
  include_loadings = TRUE,
  include_covariances = TRUE
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>discrepancy</code>	"uls" (unweighted least squares) or "ntml" (model-implied normal-theory / reweighted least squares).
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If <code>NULL</code> , infer the estimator map from <code>fit</code> .
<code>candidates</code>	"fixed" for fixed rows already present in the partable, or "all" to also enumerate absent cross-loadings and covariances.
<code>include_loadings, include_covariances</code>	Scope absent-row enumeration when <code>candidates = "all"</code> .

Value

A data frame with candidate columns and `.raw`, `.resid`, and `drop` diagnostics.

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

`noniterative_cfa_pseudo_lrt`

Pseudo-LRT between estimator-side nested non-iterative CFA fits.

Description

`fit_H1` is the less restricted model and `fit_H0` is the more restricted estimator-side fit, typically from `fit_noniterative_cfa_restricted()`. The statistic is $T_{\text{diff}} = T(H0) - T(H1)$ with an H1-anchored weighted-chi-square reference. This is an LRT-shaped discrepancy comparison; unlike ML, it can be negative for a non-minimizing estimator.

Usage

```
noniterative_cfa_pseudo_lrt(
  fit_H1,
  fit_H0,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL
)
```

Arguments

<code>fit_H1</code>	Less restricted non-iterative fit.
<code>fit_H0</code>	More restricted non-iterative fit.
<code>discrepancy</code>	"uls" (unweighted least squares) or "ntml" (model-implied normal-theory / reweighted least squares).
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If NULL, infer the estimator map from <code>fit</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

```
noniterative_cfa_scalar_invariance
```

True (lavaan-compatible) scalar-invariance test via the reference-group map.

Description

Frees the latent means. Reads the common metric off the reference group and solves the latent means by least squares through the reference loadings: $\alpha_g = (L_r' L_r)^{-1} L_r' (m_g - m_r)$, $\nu = m_r$. Scalar invariance is tested on the mean residual $d_g = (I - P_{L_r})(m_g - m_r)$ with a linearized pseudo-inverse Wald W on $df = (G-1)(p - \#factors)$ degrees of freedom. The fit must carry mean structure (fit a configural meanstructure model first) and at least two groups.

Usage

```
noniterative_cfa_scalar_invariance(  
  fit,  
  ref_group = 1,  
  discrepancy = c("uls", "ntml"),  
  gamma = c("nt", "empirical"),  
  data = NULL,  
  estimator = NULL  
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>ref_group</code>	Reference group (1-based; latent means are 0 there).
<code>discrepancy</code>	Accepted for compatibility; ignored because this wrapper only needs the delta-method covariance.
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If NULL, infer the estimator map from <code>fit</code> .

Details

Returns the latent means `alpha` (a list, one per non-reference group) with delta-method `alpha_se` / `alpha_cov`, the common intercept `nu`, the stacked residual `d`, and the test (`W`, `df`, `rank`, `p_value`).

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

`noniterative_cfa_se` *Delta-method standard errors for a non-iterative CFA fit.*

Description

Returns only `se` and `vcov`, skipping residual goodness-of-fit spectra.

Usage

```
noniterative_cfa_se(
  fit,
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If <code>NULL</code> , infer the estimator map from <code>fit</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

`noniterative_cfa_wald` *Wald test of a linear restriction $R\theta = q$ for a non-iterative CFA fit.*

Description

Exact chi-square on `nrow(R)` degrees of freedom, using the delta-method covariance. The primary nested test for these estimators.

Usage

```
noniterative_cfa_wald(
  fit,
  R,
  q,
  discrepancy = c("uls", "ntml"),
  gamma = c("nt", "empirical"),
  data = NULL,
  estimator = NULL
)
```

Arguments

<code>fit</code>	A fit from <code>fit_noniterative_cfa()</code> .
<code>R</code>	Restriction matrix (<code>k</code> x <code>n_free</code>).
<code>q</code>	Right-hand side (length <code>k</code>).
<code>discrepancy</code>	Accepted for compatibility; ignored because this wrapper only needs the delta-method covariance.
<code>gamma</code>	Fourth-moment matrix: "nt" (normal-theory, from the fitted Sigma) or "empirical" (distribution-free; requires <code>data</code>).
<code>data</code>	Raw data matrix (rows = observations), required when <code>gamma = "empirical"</code> .
<code>estimator</code>	Non-iterative estimator; see <code>fit_noniterative_cfa()</code> . If NULL, infer the estimator map from <code>fit</code> .

Note

Noniterative development is postponed. The NT default is a normal-theory comparator and does not provide structural-misspecification robustness.

<code>parameter_table</code>	<i>Assemble a lavaan-shaped parameter table (est / se / z / p / CI) from a fit and an explicit parameter covariance.</i>
------------------------------	--

Description

Estimator-agnostic: works for any magmaan fit given a `vcov` on the free parameters. Rows are the free-parameter block of the fitted partable, in theta order (by the `free` index), so `est`, `se`, `z`, and `pvalue` are aligned.

Usage

```
parameter_table(fit, vcov, level = 0.95)
```

Arguments

<code>fit</code>	A fitted magmaan model.
<code>vcov</code>	Free-parameter covariance (<code>npar x npar</code>), e.g. <code>vcov(fit, regime = "delta_nt")</code> or <code>noniterative_cfa_se(fit)\$vcov</code> .
<code>level</code>	Confidence level for the Wald CI columns.

Value

A data frame with `lhs`, `op`, `rhs`, `group`, `est`, `se`, `z`, `pvalue`, `ci.lower`, `ci.upper`.

<code>policy_fit_measures</code>	<i>Policy fit indices with misspecification-robust point corrections</i>
----------------------------------	--

Description

Computes indices on request, including an independence baseline under the same estimator and data treatment. ML and FIML use the raw-likelihood Takeuchi trace difference; ordinal and mixed DWLS use exact first-stage influence with the estimated-weight channel. Continuous ULS uses its profile trace plus covariance moment bias; ML2S-NT uses Stage-1 Gamma. No intervals are returned. RMSEA includes the multigroup factor $\sqrt{G}$. TLI uses nominal degrees of freedom and truncated corrected discrepancies; the lab misspecification family's generalized-df TLI remains a separate comparator. Residual indices pool squared residuals across groups and subtract their influence trace. DWLS indices measure misfit in the DWLS metric; ML cutoffs do not transfer.

Usage

```
policy_fit_measures(fit)
```

Arguments

<code>fit</code>	A fitted magmaanlab model retaining raw data or categorical stats.
------------------	--

Value

A data frame with `index`, `estimate` and `reason`, and a `details` attribute carrying discrepancies, traces, corrected discrepancies and degrees of freedom. Unsupported components have NA estimates and explicit reasons.

policy_inference	<i>Compose ordinary-user inference</i>
------------------	--

Description

Requests the C++ ordinary-user inference policy for the fitted estimator and data regime. Unsupported regimes remain explicit.

Usage

```
policy_inference(fit, data = NULL)
```

Arguments

fit	A fitted magmaan model.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Details

ML/FIML use observed-bread empirical-score sandwich covariance and score and likelihood-ratio tests with PEBA4 as the default reference. All-ordinal and complete mixed DWLS use exact empirical first-stage sampling influence and estimated-weight IJ covariance. Global fit-function tests and observed-Hessian/IJ nested fit-function differences use the exact weighted chi-square All reference; SB/PEBA4 comparators remain available. Fitting retains OPG NACOV weights. Reference-law studies support the DWLS default across ordinal, mixed, threshold-invariance and textbook designs; finite-sample size varies across settings. Nested DWLS score tests are unavailable.

Value

An inference bundle with availability status, covariance and test summaries where supported.

policy_mi_refit	<i>Refit one modification-index candidate from the embedded null</i>
-----------------	--

Description

Frees one candidate (or releases one equality) and refits from the fitted estimate embedded in the augmented model, replaying the fit's options. The template is an evaluation point for the native embedding map; it is never optimized.

Usage

```
policy_mi_refit(fit, row, candidate_row)
```

Arguments

fit The fitted (restricted) magmaanlab model.

row One row of `policy_modification_indices` output.

candidate_row The matching element of that output's `candidate_row` attribute.

Value

The fitted augmented model, for `policy_nested`.

`policy_modification_indices`

Policy modification indices and equality releases

Description

Thin adapter to the C++ ordinary-policy modification-index composer.

Usage

```
policy_modification_indices(fit, data = NULL, releases = TRUE)
```

Arguments

fit An estimate-only magmaanlab fit.

data Optional raw observations, otherwise the retained fitting data.

releases Include univariate equality-release score tests.

Details

ML/FIML use observed nuisance sensitivity with expected metric/EPCs and empirical likelihood-score meat (pattern scores for FIML), and all-ordinal/mixed DWLS use exact first-stage sampling rows and estimated-weight influence with observed sensitivity. Each available robust score statistic has a chi-square(1) limiting reference. Singular nuisance sensitivity or nonpositive score variance retains `numeric_failure` rows. Fits outside the policy return typed unavailable rows. No alternative inference convention is substituted.

Value

A data frame with `kind`, `lhs`, `op`, `rhs`, `group`, `test`, `statistic`, `df`, `pvalue`, `epc`, `sepc.lv`, `sepc.all` and `reason`. Internal candidate-row metadata supports native one-restriction augmentation and embedded-start refits.

policy_nested	<i>Compose ordinary-user nested inference</i>
---------------	---

Description

Requests the C++ ordinary-user nested policy for a restricted and unrestricted fit.

Usage

```
policy_nested(fit_H1, fit_H0, data = NULL)
```

Arguments

fit_H1	Unrestricted (alternative) fitted model.
fit_H0	Restricted (null) fitted model.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.

Details

ML/FIML use observed-bread empirical-score sandwich covariance and score and likelihood-ratio tests with PEBA4 as the default reference. All-ordinal and complete mixed DWLS use exact empirical first-stage sampling influence and estimated-weight IJ covariance. Global fit-function tests and observed-Hessian/IJ nested fit-function differences use the exact weighted chi-square All reference; SB/PEBA4 comparators remain available. Fitting retains OPG NACOV weights. Reference-law studies support the DWLS default across ordinal, mixed, threshold-invariance and textbook designs; finite-sample size varies across settings. Nested DWLS score tests are unavailable.

Value

A nested inference bundle with availability status and supported test summaries.

prepare_inference_data	<i>Shared continuous NTML inference geometry</i>
------------------------	--

Description

Own sample contributions and fitted NTML geometry across score, likelihood-ratio, goodness-of-fit and covariance calculations. These are methods-development interfaces for interior, random-X complete-data ML fits with affine equality constraints.

Usage

```
prepare_inference_data(fit, data = NULL,
  storage = c("auto", "casewise", "tiled"))
prepare_hypothesis(null, alternative)
inference_quadratic(object, test = c("score", "lr"),
  geometry = NULL)
inference_covariance(context, robust = TRUE)
inference_reuse(context)
```

Arguments

geometry	Observed information for nested hypotheses by default; expected information for global goodness-of-fit under the correct-model null. Either may be chosen explicitly.
fit	A continuous ML fit defining the observed-variable/group layout.
data	Fitting observations or a prepared dataset. Defaults to retained fitting data.
storage	Cache centered casewise moment rows, use tiled projection, or select case-wise storage when the contribution matrix occupies at most 64 MiB.
null, alternative	Inference snapshots created with <code>prepare_inference(fit, shared_data)</code> using the same inference dataset. Exact parameter nesting requires matching ambient parameter slots and numeric models. The null adds affine restrictions.
object	An inference snapshot (global test), or prepared hypothesis (nested test).
test	Expected-information score or ML likelihood-ratio quadratic.
context	A prepared inference snapshot.
robust	Use the observed-information empirical sandwich covariance (the default), or explicitly request inverse expected information with <code>robust = FALSE</code> .

Details

The core reuses the existing NTML U-factor construction and casewise covariance reducers. It retains structured expected-information geometry and each test's reduction separately. Global LR uses the centered empirical GOF spectrum; global score uses likelihood contributions at the fitted null point. Restricted means require additional linear and constant terms, so these spectra are not silently equated. Nested LR uses the exact H1-anchored Satorra–2000 reduction; nested score evaluates at H0.

No empirical Gamma or full U matrix is required. Casewise storage reuses one contribution matrix across fits. Tiled storage reduces global score and GOF together when the sample size is at least `df`, retaining reduced matrices; otherwise the smaller row-space spectrum is used. Tiled storage trades repeated raw-data passes for bounded contribution memory when different hypotheses need different projections. Explicit Browne-unbiased GOF requests use the existing casewise correction and may materialize contributions.

`score_spectrum` and `calibrate_quadratic` accept these quadratics. Spectrum calculations are lazy and retained; SB alone uses the trace. `inference_covariance` returns

a snapshot-tagged matrix usable with `wald_test`. Existing GOF, expected-information, expected sandwich-SE and exact empirical streaming nested-LR wrappers also reuse compatible prepared snapshots. Other `bread`/`Gamma`/`nesting` conventions retain their existing paths.

Shared NTML geometry does not yet cover FIML, ML2S, ordinal or fixed-X fits. Their existing score interfaces remain available. Native handles are local to the current R process: prepare again after serialization. Locked R environments preserve ownership; edited extracted fit lists invalidate the native caches.

Value

Prepared data/hypothesis and quadratic calls return locked environments owning native objects. Covariance returns a matrix; reuse returns construction counts, including data contributions, fit geometry, U, information and global spectra. Counts are diagnostics, not a timing guarantee.

Examples

```
## Not run:
d <- prepare_inference_data(fit1, data)
g0 <- prepare_inference(fit0, d)
g1 <- prepare_inference(fit1, d)
h <- prepare_hypothesis(g0, g1)
qs <- inference_quadratic(h, "score")
ql <- inference_quadratic(h, "lr")
calibrate_quadratic(qs, c("sb", "peba4"))
calibrate_quadratic(ql, c("sb", "peba4"))
V <- inference_covariance(g0, robust = TRUE)
inference_reuse(g0)

## End(Not run)
```

```
prepare_model
```

```
Reusable native model, data and weight preparation
```

Description

Separate structural model preparation, dataset preparation, estimation weights and estimation. Inference is requested separately.

Usage

```
prepare_model(model, ..., prototype = NULL)
augment_model_spec(model, categories)
prepare_data(model, data, kind = NULL, missing = c("error", "listwise"))
prepare_weight(data, method = c("DWLS", "WLS", "ULS", "GLS", "DLS"),
               W = NULL, full = TRUE, dls_a = 0.5)
estimate(model, data, estimator = NULL, weight = NULL,
```

```
optimizer = NULL, control = NULL, bounds = NULL,
covariance = NULL, psd = FALSE, barrier = NULL, dls_a = 0.5,
options = NULL)
```

Arguments

model	Model syntax or a model specification for prepare_model ; a specification for augment_model_spec ; a prepared model for the other functions.
...	Options passed to model_spec when model is syntax.
prototype	Data frame or named matrix supplying categorical levels and order. No empirical moments or starts are retained from it.
categories	For augment_model_spec , a list named by the ordered variables giving each variable's categories in order. Only category counts enter augmentation. Threshold starts remain missing and are initialized from each fitted dataset. The returned specification records the categories.
data	A data frame, named matrix or sample-moment list for prepare_data ; a prepared dataset for the other functions.
kind	"moments", "raw" (FIML), "ordinal" or "mixed". Default follows the model schema.
missing	Reject missing observations or delete incomplete rows for moment data. Raw FIML data retain missingness.
method	Fixed weight: ULS, GLS/NT, DWLS, WLS/ADF or DLS for complete continuous and all-ordinal data. Lower-case weight names are also accepted. Mixed data retain DWLS/WLS.
W	Optional weight matrix or per-group list in model moment order. Otherwise empirical weights are constructed from raw observations.
full	For categorical data, retain full Gamma for post-fit inference. FALSE requests only the DWLS diagonal (or no Gamma for ULS) when W is omitted. GLS, DLS, WLS and supplied ordinal weights retain full Gamma. DWLS does not invert full Gamma.
estimator	Continuous or all-ordinal ML, ULS, GLS, DWLS, WLS or DLS; FIML for raw data; DWLS or WLS for mixed data. Default follows the weight, or otherwise the data kind.
weight	Optional prepared weight from this dataset, or a shared weight name: "uls", "nt", "dwls", "adf", or "dls". Required weights are prepared internally if omitted.
dls_a	Fixed DLS mixing scalar in [0, 1]. Zero selects NT and one ADF. When a prepared weight is supplied its stored scalar governs.
optimizer, control	Existing native optimizer name and control list. A keyed data-frame control\$start supplies fit-local start hints.
options	Versioned fitting choices (preset, starts, optimizer and convergence) for unrestricted continuous ML or FIML, as in fit_model .
bounds	Explicit lower/upper bound list in free-parameter order. Not supported by the prepared FIML path.

covariance	Model covariance policy: "unrestricted", "psd" or "barrier". NULL follows psd . Mixed models support unrestricted and PSD fitting; barriers are deferred.
psd	Compatible spelling for covariance = "psd". Explicit contradictory requests are rejected.
barrier	Options for barrier fitting: a list with target ("joint" or "determinacy") and weight (default 0.25). The weight is finite and non-negative; zero disables the penalty.

Details

Rebuild handles after serialization and in each worker. A new dataset requires new data and weight preparation, but reuses the model. Category levels, group order, observable order and mean-structure convention must remain compatible. Group labels must be declared explicitly. Covariances supplied as sample moments use divisor N .

The covariance policy changes final model fitting without repairing supplied moments. A barrier minimizes $f - \text{weight} / N * P$, with **fmin** retaining the unpenalized discrepancy and **penalty\$penalized_fmin** recording the fitted objective. Positive-weight barriers replace additional bounds. Their general sampling/inference contract is not yet validated. Ordinal ML fits associations with Stage-1 thresholds held fixed; it does not provide Gaussian likelihood or generic post-fit sampling inference.

Continuous diagonal weight blocks use owning numeric vectors of entries of W , not square roots; full blocks remain matrices. Supplied W accepts either form, or a list of blocks. Prepared native handles are process-local: serialize W and prepare a new handle after loading it.

FIML retains its missingness pack but does not compute saturated $H1$ during estimation unless versioned fitting options require it. Categorical inference needs full Gamma; obtain it separately with `prepare_weight(data, "DWLS", full = TRUE)$stats` and pass it to an existing explicit inference function, without refitting.

Versioned fitting options use the same configured engines as `fit_model()` for continuous ML/FIML and ordinal/complete mixed DWLS with unrestricted covariance, including `options = list(preset = "lavaan-0.7.2")`.

The interface covers the estimators listed above. ML2S raises a typed `magmaan_unsupported_estimator` error. Two-level, FC-SEM, SAM and frontier methods continue using their existing entry points, which remain supported without deprecation warnings. Categorical weight preparation currently recomputes stage-one moments in the existing core builder; fitting itself reuses the prepared objects.

Value

augment_model_spec returns a model specification with threshold and scale rows added once. Preparation returns immutable, process-local handles. A prepared model exposes its specification; a weight exposes W for continuous data and **stats** for categorical inference. Estimation returns the existing `magmaan_fit` list with convergence and admissibility diagnostics.

Deletion provenance

Dataframe helpers, prepared data and fitted lab objects retain a read-only **deletion** record. Its **policy** identifies listwise, error, FIML or pairwise row handling; **known** indicates whether supplied-row counts are available. **per_group** contains **group**, **rows_supplied**, **rows_used**, **rows_deleted**, and exclusive reason counts **missing_model_values**, **missing_group_label**, **unknown_group_label**, and **missing_cluster_id**. Reasons take precedence in that order: group labels, cluster IDs, model values. Rows without an assignable group appear in a separate **NA** group row. Single-level routes still refuse missing or unknown group labels; two-level listwise preparation deletes missing group or cluster IDs. FIML and ML2S retain incomplete rows without listwise deletion; existing native refusals remain. Summary-only inputs without retained provenance report unknown supplied/deleted counts, rather than claiming that no earlier deletion occurred.

<code>refit_from_null</code>	<i>Refit a larger model from a restricted estimate</i>
------------------------------	--

Description

Embeds the restricted estimate using the native, moment-verified nesting map, then replays the larger model's estimator and fitting options with that explicit start. Neither input is modified.

Usage

```
refit_from_null(fit_H1, fit_H0)
```

Arguments

<code>fit_H1</code>	The larger fitted model.
<code>fit_H0</code>	The restricted fitted model, using the same data and estimator.

Details

Supports single-level ML, FIML and all-ordinal DWLS, ULS and WLS fits from `fit_model()` or `estimate()`. FIML requires retained observations; complete-data and ordinal routes reuse retained sufficient statistics. This explicit lab operation returns the new endpoint even when its verdict fails; callers must inspect convergence diagnostics. Fit warnings are emitted as for any explicit user-requested fit.

Value

A new estimate-only `magmaan_fit`, with its own convergence diagnostics and native verdict.

robust_nested_lrt	<i>Robust nested-model likelihood-ratio tests.</i>
-------------------	--

Description

Compares a restricted model `fit_H0` against the less-restricted superset `fit_H1`. `method = "restriction_map"` uses Satorra's streaming low-rank reduction of the asymptotic weighted-chi-square distribution. The `"lavaan_sb2001"` and `"lavaan_sb2010"` options mirror lavaan's compatibility difference-test approximations.

Usage

```
robust_nested_lrt(
  fit_H1,
  fit_H0,
  data = NULL,
  gamma = c("empirical", "unbiased", "NT", "both"),
  method = c("restriction_map", "lavaan_sb2001", "lavaan_sb2010"),
  A.method = c("exact", "delta"),
  computation = c("streaming", "materialized", "dense"),
  convention = c("magmaan", "lavaan"),
  ud_method = c("2000", "2001"),
  h1_reference_regularization = NULL,
  weight = NULL
)

nestedTest(
  fit_H1,
  fit_H0,
  data = NULL,
  gamma = c("empirical", "unbiased", "NT", "both"),
  method = c("satorra.2000", "satorra.bentler.2001", "satorra.bentler.2010",
    "restriction_map", "lavaan_sb2001", "lavaan_sb2010"),
  A.method = c("exact", "delta"),
  computation = c("streaming", "materialized", "dense"),
  convention = c("magmaan", "lavaan"),
  ud_method = c("2000", "2001"),
  h1_reference_regularization = NULL,
  weight = NULL
)
```

Arguments

<code>fit_H1</code>	Less-restricted fitted magmaan model, ordinarily returned by <code>fit_model()</code> .
<code>fit_H0</code>	More-restricted fit, with paths dropped, fixed or constrained. Interior reparameterizations through moments are supported for ML/FIML.

<code>data</code>	Raw complete data for complete-data fits: either a data.frame whose columns include the observed variables of <code>fit_H1</code> (single-group case), or a list of per-group matrices in the same block order the fit was built with. Ordinal DWLS/WLS pairs take the <code>magmaan_ordinal_data</code> object used for fitting. FIML pairs use <code>fit_H1\$raw_data</code> and do not accept <code>data</code> .
<code>gamma</code>	"empirical" (default, empirical Gamma-hat), "unbiased" (the Browne/Du-Bentler finite-sample correction for complete-data ML), "both" (both spectra in one shared streaming pass; the unbiased spectrum is returned as <code>eigenvalues_unbiased</code> ; complete-data ML restriction-map pairs with <code>computation = "streaming"</code> only), or "NT" (normal-theory sanity-check path where all eigenvalues collapse to 1).
<code>method</code>	Historical lavaan labels (" <code>satorra.2000</code> ", " <code>satorra.bentler.2001</code> ", " <code>satorra.bentler.2010</code> ") or the canonical robust nested-LRT method names.
<code>A.method</code>	"exact" (default, exact parameter-nesting restriction) or "delta" (lavaan-style covariance/moment Jacobian column-space restriction). FIML pairs with nonlinear equality constraints use the local tangent restriction map even when "exact" is requested, because no global affine exact map exists. For lavaan's default FIML <code>lavTestLRT(method = "satorra.2000")</code> comparison, use "delta".
<code>computation</code>	"streaming" (default, projects casewise empirical Gamma contributions before crossproducts), "materialized" (forms the full empirical Gamma block before reducing), or "dense" (forms the full q-by-q Gamma and U matrices and eigendecomposes the q-by-q product, as standard SEM software does). FIML lavaan-convention tests use the same choices with the retained saturated EM covariance: "streaming" applies WLS.V to restriction columns and contracts that covariance without an additional full Gamma; "materialized" forms Gamma explicitly and "dense" is a diagnostic oracle.
<code>convention</code>	"magmaan" with method omitted calls the ML/DWLS policy composer. With an explicit method it retains the estimator-specific Satorra comparator. "lavaan" uses lavaan's public <code>lavTestLRT(method = "satorra.2000")</code> convention for FIML/ML2S missing-data pairs. For FIML with empirical Gamma, it combines the saturated EM sandwich covariance, expected WLS.V at the fitted larger model, and observed parameter information, matching lavaan MLR with its default information settings. ML2S retains its Stage-2 weight and raw-moment Gamma convention. Both use "delta" as the default <code>A.method</code> when the caller does not specify one.
<code>ud_method</code>	Estimator of the U_D difference matrix whose $U_D * \text{Gamma}$ eigenvalues drive the scaled statistic: "2000" (default, Satorra 2000 restriction map from the H1 fit) or "2001" (Satorra-Bentler 2001 $U_D = U_0 - U_1$, the difference of the two single-model projectors; <code>semTests::ugamma_nested(., "2001")</code>). "2001" is implemented for FIML and ML2S fits only; its spectrum can carry negative eigenvalues. "2000" accepts dropped/fixed paths through the shared exact null embedding.
<code>h1_reference_regularization</code>	FIML restriction-map tests only. NULL or FALSE keeps the raw saturated-H1 reference. TRUE applies the frontier condition-cap defaults. A list

may set `condition_max`, `min_eigenvalue`, and optional `covariance / information` sublists.

weight For ordinal nested tests, "DWLS", "WLS", or "ULS". For continuous WLS, the explicit fitting weight matrix (or per-group list of matrices). Continuous ULS/GLS rebuild their own weights and reject this argument.

Details

With `convention = "magmaan"` and method omitted, `robust_nested_lrt()` calls `policy_nested()` for complete-data ML and all-ordinal DWLS. Unsupported components raise a condition of class `magmaan_unsupported_inference`, with `reason` and `detail` fields. Explicit method choices and `nestedTest()` are Satorra/lavaan compatibility routes and retain their documented formulas.

Value

A list of class `magmaan_nested_test`. With `gamma = "both"`, `eigenvalues` and the scalar test summaries use empirical Gamma, while `eigenvalues_unbiased` contains the auxiliary unbiased spectrum.

<code>robust_ordinal_ij</code>	<i>All-ordinal estimated-weight IJ covariance</i>
--------------------------------	---

Description

Composes observed-bread covariance with the estimated-weight influence. The exact comparator changes sampling rows and their contribution to weight movement; fitting continues to use OPG NACOV weights.

Usage

```
robust_ordinal_ij(fit, ordinal_stats, weight = "", first_stage = "opg")
```

Arguments

fit An all-ordinal fitted lab model.

ordinal_stats The ordinal statistics used for fitting.

weight Computational weight; empty reuses the fit's weight.

first_stage "opg" (default) or the complete-data empirical score-Jacobian comparator "exact".

Value

A list containing `vcov`, `se`, `df` and `chisq_standard`. The exact comparator also returns per-group `sampling_moment_influence` matrices and `sampling_gamma` (the cross-product of the rows divided by group sample size).

sam

*Structural after measurement estimation***Description**

Composes separate measurement and structural fitting using the selected mapping and covariance convention.

Usage

```
sam(model, data, method = c("local", "global"), mapping = c("ml", "gls", "uls"),
     se = c("twostep", "twostep.robust", "standard", "none"), lambda_correction = TRUE,
     alpha_correction = 0L, meanstructure = FALSE, mm_optimizer = "nlopt-lbfgs", struc_optimizer =
     mm_control = NULL, struc_control = NULL, raw_data = NULL, missing = c("listwise",
     "error"), ...)
```

Arguments

model	Model syntax or a model specification; prepared objects are accepted only by their corresponding interfaces.
data	Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
method	Method selector; see the available choices in the usage signature and details.
mapping	SAM measurement-to-structural mapping: ml, gls or uls.
se	Standard-error convention. <code>fit_model</code> is estimate-only and accepts only "none"; SAM and <code>est_change</code> expose the choices shown in their signatures.
lambda_correction	Apply the SAM loading correction.
alpha_correction	SAM alpha-correction mode.
meanstructure	Whether to include intercepts and means; <code>model_spec</code> also accepts the default convention.
mm_optimizer	Optimizer for the SAM measurement model.
struc_optimizer	Optimizer for the SAM structural model.
mm_control	Control list for the SAM measurement fit.
struc_control	Control list for the SAM structural fit.
raw_data	Optional raw observations needed for robust SAM inference.
missing	Missing-data handling: listwise deletion, rejection, or pairwise moments where supported. FIML retains missing observations.
...	Additional options forwarded to model construction or the underlying named operation.

Value

A fitted SAM result with measurement and structural components.

Note

The SE/type defaults standardize SAM or case-change diagnostics. They do not assert a generic misspecification-robust inference law; explicit robust or estimated-weight choices remain available within their supported scope.

score_flip_test	<i>Multiplier score test for affine nested ML/FIML models.</i>
-----------------	--

Description

Calibrates the restrictions separating `fit_H0` from `fit_H1` by random mean-zero, unit-variance multipliers on individual likelihood-score contributions. Rademacher signs are the default and support the basic, nuisance-effective, and flip-specifically standardized references. Mammen, tunable two-point, Gaussian, and centered-exponential weights are experimental diagnostics for the nuisance-effective reference.

Usage

```
score_flip_test(
  fit_H1,
  fit_H0,
  data = NULL,
  n_flips = 999L,
  seed = 1,
  calibration = c("effective", "all", "effective-standardized", "asymptotic"),
  multiplier = c("rademacher", "mammen", "two-point", "gaussian", "centered-exponential"),
  two_point_skewness = 1,
  center_multiplier_scores = FALSE,
  multiplier_studentization = c("none", "weighted-meat"),
  sensitivity = c("observed", "expected")
)
```

Arguments

<code>fit_H1</code>	Less-restricted complete-data ML or direct-FIML fit, or a <code>magmaan_model_spec</code> for the less-restricted model. Supplying the model avoids fitting H1 because a score test uses only H1's tangent and H0's estimates.
<code>fit_H0</code>	More-restricted nested fit with paths dropped, fixed or constrained, using the same estimator and observations. Interior moment reparameterizations are evaluated at a common null point.
<code>data</code>	Raw fitting data for complete-data ML. Direct FIML uses the observed-data sample stored on both fits, so this argument must be omitted.

<code>n_flips</code>	Number of random multiplier transformations.
<code>seed</code>	Non-negative deterministic integer seed.
<code>calibration</code>	Which randomization references to compute. "effective" skips the basic and flip-specific covariance calculations; "effective-standardized" adds the standardized reference; "all" preserves the original three-reference result. "asymptotic" draws no multipliers and is used by <code>nested_score_test()</code> .
<code>multiplier</code>	Multiplier distribution. Non-Rademacher resampling requires <code>calibration = "effective"</code> .
<code>two_point_skewness</code>	Non-negative third moment for <code>multiplier = "two-point"</code> . The corresponding fourth moment is the minimum possible value, $1 + \text{two_point_skewness}^2$; zero gives a Rademacher law and one gives Mammen's law.
<code>center_multiplier_scores</code>	Whether to center effective-score contributions within information strata before applying multipliers.
<code>multiplier_studentization</code>	Optional "weighted-meat" bootstrap-t diagnostic using the per-draw meat $\sum(w_i^2 u_i u_i')$. "none" retains the expected-information metric.
<code>sensitivity</code>	Nuisance sensitivity (bread). "expected" preserves the original pattern-conditional normal-theory construction; "observed" uses the realized likelihood Hessian for the nuisance projection and supplies the pseudo-true/MAR correction when information equality need not hold. The score quadratic retains the expected metric.

Value

A list of class `magmaan_score_flip_test`. In addition to the three p-values it includes the mean-scaled and exact-mixture score references, the direct sandwich-studentized score statistic and p-value, flip-variance displacement diagnostics, optional multiplier-studentized output, and elapsed seconds for setup, resampling, flip standardization, and the asymptotic comparators.

<code>score_tests</code>	<i>Post-fit parameter diagnostics: score_tests</i>
--------------------------	--

Description

Computes explicit release or restriction diagnostics. LR-type variants use the supplied observations for robust spectral inference. Unsupported candidate regimes are reported explicitly.

Usage

```
score_tests(fit, data = NULL, ...)
```

Arguments

- `fit` A fitted magmaan model.
- `data` Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
- `...` Additional options forwarded to model construction or the underlying named operation.

Value

A table of candidate parameter diagnostics.

<code>score_tests_lrt</code>	<i>Post-fit parameter diagnostics: score_tests_lrt</i>
------------------------------	--

Description

Computes explicit release or restriction diagnostics. LR-type variants use the supplied observations for robust spectral inference. Unsupported candidate regimes are reported explicitly.

Usage

```
score_tests_lrt(fit, data, candidates = c("loadings", "intercepts", "all"), method = c("restriction", "lavaan_sb2001", "lavaan_sb2010"), gamma = c("empirical", "NT"))
```

Arguments

- `fit` A fitted magmaan model.
- `data` Observed data frame, named matrix or sample-moment list compatible with the model. FIML requires raw observations.
- `candidates` Candidate parameter rows to release or test: all or the named subset supported by the operation.
- `method` Method selector; see the available choices in the usage signature and details.
- `gamma` Moment covariance convention used for the nested score diagnostic.

Value

A table of candidate parameter diagnostics.

scores

*Reusable score ingredients and explicit inference***Description**

Prepare an immutable native inference snapshot, obtain scores and geometry, and choose projection, calibration or resampling separately. These methods-developer interfaces support complete-data ML, direct FIML and the unregularized fixed-normal-theory ML2S global score construction.

Usage

```
prepare_inference(fit, data = NULL)
scores(object, data = NULL, space = c("parameter", "saturated"))
score_components(object, data = NULL, H1 = NULL,
  sensitivity = c("observed", "expected", "observed-h1",
    "observed-shrink-light", "observed-shrink-sqrt"),
  metric = c("expected", "observed", "observed-h1"))
score_components_from_matrices(score, rows, sensitivity, metric = sensitivity,
  nuisance = matrix(0, length(score), 0L), directions = diag(length(score)),
  influence_rows = FALSE)
project_scores(components, retain_rows = FALSE, center = FALSE)
score_quadratic(score, metric, meat)
score_spectrum(projected)
score_sandwich(projected)
resample_scores(projected, n_flips = 999L, seed = 1,
  multiplier = c("rademacher", "mammen", "two-point", "gaussian",
    "centered-exponential"), two_point_skewness = 1)
quadratic_reference(statistic, df, eigenvalues = NULL)
calibrate_quadratic(object, methods = "peba4")
inference_information(context, type = c("observed", "expected"))
parameter_covariance(context, information, meat = NULL)
wald_test(object, R, vcov, q = NULL)
```

Arguments

fit	A fitted magmaan model.
data	Fitting raw data for complete ML, optionally a prepared dataset. FIML and ML2S use the raw observations stored on the fit.
object, context	A fit or prepared inference snapshot for score construction; a projected score or quadratic reference for calibration; a fit, snapshot or estimate vector for Wald inference.
space	Parameter or saturated-moment likelihood coordinates. ML2S influence contributions are available through <code>score_components</code> .
H1	Optional unrestricted model specification or fit for affine nested ML/FIML scores. No unrestricted fit is required. Null selects global GOF.

sensitivity, metric	Nuisance sensitivity and quadratic metric choices. For supplied-matrix score_quadratic , metric is a positive-definite matrix.
rows, nuisance, directions, influence_rows	Supplied casewise rows, nuisance and tested direction matrices, and whether the rows are influence contributions rather than likelihood scores. Likelihood rows must sum to the observed score.
components, projected	Reusable objects produced by the preceding stage.
retain_rows	Retain projected rows for multiplier resampling.
center	Globally center covariance rows. Does not change the observed score. This is not within-pattern centering.
score	An observed score or contrast vector in total-likelihood units.
meat	Symmetric score covariance in total units. Required for score_quadratic . For parameter_covariance , NULL derives empirical meat from retained fitting data; explicit "model" requests inverse information.
statistic, df, eigenvalues	A precomputed quadratic statistic, nominal positive integer degrees of freedom, and optional spectrum including zero weights. Negative entries within the projected-score roundoff tolerance are clipped to zero; materially negative entries raise a NumericIssue error.
methods	Case-insensitive names std , sb , ss , mv , scaled_f , all , pall , peba<k> , eba<k> , with integer $k \geq 1$ (up to the C++ integer limit). Output names are lower case. Calibration calls the C++ FMG routine. EBA/pEBA use blocks of size $\text{ceiling}(\text{df} / k)$ in the retained spectrum, including zero weights. When $k \geq \text{df}$, EBA equals All and pEBA equals pAll.
n_flips, seed, multiplier, two_point_skewness	Explicit multiplier simulation settings. Reuses the projected metric factorization and retained rows.
type	Expected Fisher or realized observed information.
information	Previously computed total parameter information.
R, q, vcov	Wald restriction matrix, target vector (zero by default), and previously computed parameter covariance.

Details

The preparation snapshot owns its native model/data/parameter state. It is process-local; reprepare after serialization or changes to the model or sample. New geometry never draws multipliers or computes p-values. Projection retains a Cholesky factor; spectrum construction is separate and its result can be reused for many calibrations. SB-only calibration of projected scores uses a trace without a spectrum. PSD meat is allowed for mixture inference; the separate sandwich operation requires full rank.

Likelihood scores have the sign of the log-likelihood derivative and summed normalization. ML2S instead exposes the observed Stage-2 score separately from its propagated Stage-1 influence rows: those rows need not sum to the observed score. Global and nested targets

use different coordinate spaces. Geometry support restrictions follow the corresponding legacy score routines.

The snapshot can also be passed to `fmg_tests`, `fmg_nested`, and `robust_nested_lrt`. Their native fit context is reused; these legacy procedures still compute their own method-specific geometry. Save an existing LR/GOF spectrum with `quadratic_reference` to recalibrate without repeating that computation. Information and covariance preparation for Wald currently support ML/FIML; ML2S uses its existing two-stage covariance interface.

Value

Locked environments hold prepared snapshots, components, projections and reference laws. Scores and multiplier summaries are lists, information and covariance are matrices, and calibration returns a data frame. Wald returns the existing statistic, df and p-value list.

<code>sim_model_batch</code>	<i>Calibrate and draw model-implied continuous data in one call.</i>
------------------------------	--

Description

Calibrate and draw model-implied continuous data in one call.

Usage

```
sim_model_batch(
  fit_or_partable,
  n,
  reps,
  seed_base,
  theta = NULL,
  generator = "normal",
  df = 5,
  mixture_weights = NULL,
  mixture_scale_multipliers = NULL,
  contamination_probability = 0.05,
  contamination_scale_multiplier = 3,
  slash_q = 5,
  cholesky_jitter = 0
)
```

Arguments

<code>fit_or_partable</code>	A magmaan fit object or partable.
<code>n</code>	Number of observations per replication.
<code>reps</code>	Number of replications.
<code>seed_base</code>	Base RNG seed.

<code>theta</code>	Optional free-parameter vector when <code>fit_or_partable</code> is a partable.
<code>generator</code>	Continuous generator name; default "normal".
<code>df</code>	Degrees of freedom for Student t generation.
<code>mixture_weights</code>	Mixture component probabilities.
<code>mixture_scale_multipliers</code>	Positive component scale multipliers.
<code>contamination_probability</code>	Probability of a contaminated observation.
<code>contamination_scale_multiplier</code>	Scale multiplier for contamination.
<code>slash_q</code>	Positive exponent for slash generation.
<code>cholesky_jitter</code>	Non-negative diagonal jitter used for factorization.

`sim_model_calibrate` *Calibrate model-implied continuous data generation.*

Description

Calibrate model-implied continuous data generation.

Usage

```
sim_model_calibrate(fit_or_partable, theta = NULL)
```

Arguments

<code>fit_or_partable</code>	A magmaan fit object or partable.
<code>theta</code>	Optional free-parameter vector when <code>fit_or_partable</code> is a partable.

`sim_model_draw` *Draw model-implied continuous data from a calibration object.*

Description

Draw model-implied continuous data from a calibration object.

Usage

```

sim_model_draw(
  calibration,
  n,
  reps,
  seed_base,
  generator = "normal",
  df = 5,
  mixture_weights = NULL,
  mixture_scale_multipliers = NULL,
  contamination_probability = 0.05,
  contamination_scale_multiplier = 3,
  slash_q = 5,
  cholesky_jitter = 0
)

```

Arguments

<code>calibration</code>	A calibration from <code>sim_model_calibrate()</code> .
<code>n</code>	Number of observations per replication.
<code>reps</code>	Number of replications.
<code>seed_base</code>	Base RNG seed.
<code>generator</code>	Continuous generator name; default "normal".
<code>df</code>	Degrees of freedom for Student t generation.
<code>mixture_weights</code>	Mixture component probabilities.
<code>mixture_scale_multipliers</code>	Positive component scale multipliers.
<code>contamination_probability</code>	Probability of a contaminated observation.
<code>contamination_scale_multiplier</code>	Scale multiplier for contamination.
<code>slash_q</code>	Positive exponent for slash generation.
<code>cholesky_jitter</code>	Non-negative diagonal jitter used for factorization.

<code>sim_ordcorr_batch</code>	<i>Calibrate and draw ordinal/continuous Pearson-code data in one call.</i>
--------------------------------	---

Description

Calibrate and draw ordinal/continuous Pearson-code data in one call.

Usage

```

sim_ordcorr_batch(
  target_corr,
  marginals,
  n,
  reps,
  seed_base,
  metric = "polychoric",
  max_bisection_iter = 80L,
  calibration_tol = 1e-08,
  rho_bound = 0.999,
  matrix_repair = "none",
  matrix_repair_min_eigenvalue = 1e-08,
  cholesky_jitter = 0
)

```

Arguments

<code>target_corr</code>	Target observed correlation matrix.
<code>marginals</code>	List of marginal category probabilities, or NULL entries for continuous variables.
<code>n</code>	Number of observations per replication.
<code>reps</code>	Number of replications.
<code>seed_base</code>	Base RNG seed.
<code>metric</code>	Correlation metric used for the target; default "polychoric".
<code>max_bisection_iter</code>	Maximum iterations for each correlation inversion.
<code>calibration_tol</code>	Absolute tolerance for correlation calibration.
<code>rho_bound</code>	Upper bound on the absolute latent correlation.
<code>matrix_repair</code>	Correlation-matrix repair strategy.
<code>matrix_repair_min_eigenvalue</code>	Minimum eigenvalue requested by repair.
<code>cholesky_jitter</code>	Non-negative diagonal jitter used for factorization.

`sim_ordcorr_calibrate`

Calibrate ordinal/continuous Pearson-code correlation generation.

Description

Calibrate ordinal/continuous Pearson-code correlation generation.

Usage

```
sim_ordcorr_calibrate(
  target_corr,
  marginals,
  metric = "polychoric",
  max_bisection_iter = 80L,
  calibration_tol = 1e-08,
  rho_bound = 0.999,
  matrix_repair = "none",
  matrix_repair_min_eigenvalue = 1e-08
)
```

Arguments

<code>target_corr</code>	Target observed correlation matrix.
<code>marginals</code>	List of marginal category probabilities, or NULL entries for continuous variables.
<code>metric</code>	Correlation metric used for the target; default "polychoric".
<code>max_bisection_iter</code>	Maximum iterations for each correlation inversion.
<code>calibration_tol</code>	Absolute tolerance for correlation calibration.
<code>rho_bound</code>	Upper bound on the absolute latent correlation.
<code>matrix_repair</code>	Correlation-matrix repair strategy.
<code>matrix_repair_min_eigenvalue</code>	Minimum eigenvalue requested by repair.

<code>sim_ordcorr_draw</code>	<i>Draw ordinal/continuous Pearson-code data from a calibration object.</i>
-------------------------------	---

Description

Draw ordinal/continuous Pearson-code data from a calibration object.

Usage

```
sim_ordcorr_draw(calibration, n, reps, seed_base, cholesky_jitter = 0)
```

Arguments

<code>calibration</code>	A calibration from <code>sim_ordcorr_calibrate()</code> .
<code>n</code>	Number of observations per replication.
<code>reps</code>	Number of replications.
<code>seed_base</code>	Base RNG seed.
<code>cholesky_jitter</code>	Non-negative diagonal jitter used for factorization.

`sim_ordcorr_mg_batch` *Calibrate and draw grouped ordinal/continuous data in one call.*

Description

Calibrate and draw grouped ordinal/continuous data in one call.

Usage

```
sim_ordcorr_mg_batch(
  target_corrs,
  marginals,
  n,
  reps,
  seed_base,
  group_labels = NULL,
  metric = "polychoric",
  max_bisection_iter = 80L,
  calibration_tol = 1e-08,
  rho_bound = 0.999,
  matrix_repair = "none",
  matrix_repair_min_eigenvalue = 1e-08,
  cholesky_jitter = 0
)
```

Arguments

<code>target_corrs</code>	List of target correlation matrices, one per group.
<code>marginals</code>	List of group marginal lists.
<code>n</code>	Number of observations per replication.
<code>reps</code>	Number of replications.
<code>seed_base</code>	Base RNG seed.
<code>group_labels</code>	Optional group labels.
<code>metric</code>	Correlation metric used for the target; default "polychoric".
<code>max_bisection_iter</code>	Maximum iterations for each correlation inversion.
<code>calibration_tol</code>	Absolute tolerance for correlation calibration.
<code>rho_bound</code>	Upper bound on the absolute latent correlation.
<code>matrix_repair</code>	Correlation-matrix repair strategy.
<code>matrix_repair_min_eigenvalue</code>	Minimum eigenvalue requested by repair.
<code>cholesky_jitter</code>	Non-negative diagonal jitter used for factorization.

```
sim_ordcorr_mg_calibrate
```

Calibrate grouped ordinal/continuous Pearson-code generation.

Description

Calibrate grouped ordinal/continuous Pearson-code generation.

Usage

```
sim_ordcorr_mg_calibrate(  
  target_corrs,  
  marginals,  
  group_labels = NULL,  
  metric = "polychoric",  
  max_bisection_iter = 80L,  
  calibration_tol = 1e-08,  
  rho_bound = 0.999,  
  matrix_repair = "none",  
  matrix_repair_min_eigenvalue = 1e-08  
)
```

Arguments

<code>target_corrs</code>	List of target correlation matrices, one per group.
<code>marginals</code>	List of group marginal lists.
<code>group_labels</code>	Optional group labels.
<code>metric</code>	Correlation metric used for the target; default "polychoric".
<code>max_bisection_iter</code>	Maximum iterations for each correlation inversion.
<code>calibration_tol</code>	Absolute tolerance for correlation calibration.
<code>rho_bound</code>	Upper bound on the absolute latent correlation.
<code>matrix_repair</code>	Correlation-matrix repair strategy.
<code>matrix_repair_min_eigenvalue</code>	Minimum eigenvalue requested by repair.

`sim_ordcorr_mg_draw` *Draw grouped ordinal/continuous data from a calibration object.*

Description

Draw grouped ordinal/continuous data from a calibration object.

Usage

```
sim_ordcorr_mg_draw(calibration, n, reps, seed_base, cholesky_jitter = 0)
```

Arguments

<code>calibration</code>	A calibration from <code>sim_ordcorr_calibrate()</code> .
<code>n</code>	Number of observations per replication.
<code>reps</code>	Number of replications.
<code>seed_base</code>	Base RNG seed.
<code>cholesky_jitter</code>	Non-negative diagonal jitter used for factorization.

`sim_ordcorr_mg_summary_calibrate`
Rebuild grouped ordinal-correlation calibrations from summary pieces.

Description

Rebuild grouped ordinal-correlation calibrations from summary pieces.

Usage

```
sim_ordcorr_mg_summary_calibrate(  
  latent_corrs,  
  kinds,  
  thresholds,  
  group_labels = NULL,  
  metric = "polychoric",  
  matrix_repair = "none",  
  matrix_repair_min_eigenvalue = 1e-08  
)
```

Arguments

<code>latent_corrs</code>	List of latent Gaussian correlation matrices.
<code>kinds</code>	Integer variable-type codes from an existing calibration.
<code>thresholds</code>	List of threshold lists, one per group.
<code>group_labels</code>	Optional group labels.
<code>metric</code>	Correlation metric used for the target; default "polychoric".
<code>matrix_repair</code>	Correlation-matrix repair strategy.
<code>matrix_repair_min_eigenvalue</code>	Minimum eigenvalue requested by repair.

`sim_ordcorr_summary_calibrate`

Rebuild an ordinal-correlation calibration from latent summary pieces.

Description

Rebuild an ordinal-correlation calibration from latent summary pieces.

Usage

```
sim_ordcorr_summary_calibrate(
  latent_corr,
  kinds,
  thresholds,
  metric = "polychoric",
  matrix_repair = "none",
  matrix_repair_min_eigenvalue = 1e-08
)
```

Arguments

<code>latent_corr</code>	Latent Gaussian correlation matrix.
<code>kinds</code>	Integer variable-type codes from an existing calibration.
<code>thresholds</code>	Threshold list from an existing calibration.
<code>metric</code>	Correlation metric used for the target; default "polychoric".
<code>matrix_repair</code>	Correlation-matrix repair strategy.
<code>matrix_repair_min_eigenvalue</code>	Minimum eigenvalue requested by repair.

standardized	<i>Standardized parameter estimates</i>
---------------------	---

Description

Standardizes parameter estimates and propagates the supplied covariance.

Usage

```
standardized(fit, vcov, type = c("all", "lv"))
```

Arguments

fit	A fitted magmaan model.
vcov	Explicit covariance matrix of free parameters, in theta order.
type	Standardization: all variables or latent variables only.

Value

A table of standardized parameter estimates and uncertainty.

standardized_rows	<i>Row-level standardized estimates</i>
--------------------------	---

Description

Includes fixed markers, group-specific transformations and ordinal latent-response residual variances. Standard errors use the delta method. Defined parameters have missing values because their expressions have no specified standardized scale.

Usage

```
standardized_rows(fit, vcov)
```

Arguments

fit	A lab fit.
vcov	Explicit covariance in the fit's free-parameter coordinates.

Value

A list of `std.lv`, `std.lv.se`, `std.all` and `std.all.se` vectors in partable row order.

```
structural_identification
```

Check structural identification

Description

Checks generic local identification without data. A prepared model reuses its construction-time report. For an ordinal specification, provide a prototype to declare categories; prototype values never enter the check. Models with unsupported moment maps or constraints are returned as unchecked. Lab fitting remains available for every status; unidentified fits fail their convergence verdict. No constraints are added by this check.

Usage

```
structural_identification(model, prototype = NULL)
```

Arguments

model A model specification, lavaan syntax, or a prepared model.
prototype Optional category-schema prototype for ordinal specifications.

Value

A list with **status** (identified, unidentified, unchecked), **reason**, **map**, reduced **n_parameters**, **n_moments**, **counting_rule**, **rank**, **n_points**, **null_tolerance**, **identified_tolerance**, **min_relative_singular_values** (one per evaluated point), and **smallest_singular_values** at the deciding point. **null_directions** has one column per free direction, with parameter labels as row names; **null_direction_text** describes the coefficients. **direction_types** and **direction_factors** classify each column as scale, location, rotation or deficit; **suggested_fixes** gives matching advice. **gauge_dimension** and **deficit_dimension** partition the nullity. Fixed matrix entries and linear equalities are enforced before testing gauge membership. Cached directions are evaluated at the deciding data-free probe (**directions_at_estimate** = FALSE). This checks generic local identification, not global uniqueness or weak identification at a fitted point.

```
vcov.magmaan_fit
```

Parameter covariance under an explicit statistical contract

Description

Selects existing C++ covariance primitives without changing the fitted model or the ordinary-user inference policy.

Usage

```
## S3 method for class 'magmaan_fit'
vcov(object, regime = NULL, data = NULL, ...)
```

Arguments

object	A fitted <code>magmaanlab</code> model.
regime	Covariance formula; see Details. <code>NULL</code> preserves the estimator-specific historical default.
data	Raw observations for empirical sandwiches or delta-method covariances. Defaults to the observations retained by the fit. FIML always uses its retained observations and rejects a replacement.
...	Reserved for generic compatibility.

Details

information_expected, information_observed Inverse expected information or observed Hessian information, respectively. Available for complete-data ML and FIML; neither includes empirical sandwich meat.

sandwich_expected, sandwich_observed Empirical-meat sandwiches with expected/Gauss-Newton or observed-Hessian bread. Available for continuous iterative fits and categorical fits where the corresponding core primitive supports the estimator, and for FIML. Categorical covariances use the retained categorical statistics. These are fixed-weight covariances.

sandwich_ij The estimated-weight (infinitesimal-jackknife) sandwich for all-ordinal DWLS, WLS and ULS fits: observed bread, Stage-1 threshold and polychoric influence, and the influence of the estimated weight. It is the ordinary DWLS policy covariance.

delta_nt, delta_empirical Non-iterative CFA delta-method covariances using normal-theory or empirical moment covariance.

stored The covariance already computed by a SAM fit.

Unsupported combinations raise an error. Defaults are **sandwich_observed** for ML and FIML and **sandwich_ij** for categorical and estimated-weight continuous GLS/WLS fits. IJ errors when the estimator's influence law is unsupported. Non-iterative CFA retains **delta_nt**: this postponed development area uses a documented normal-theory comparator. SAM retains **stored** as a diagnostic. Shared barrier compositions and pairwise MCAR fits reject this interface pending their own sampling/inference contracts.

Legacy **model** selects the historical default. Legacy **robust** selects **sandwich_observed**, except for categorical and GLS/WLS fits (**sandwich_ij**), non-iterative CFA (**delta_empirical**) and SAM (**stored**). The model alias retains its historical comparator; robust selects the empirical/IJ route described above. Prefer the explicit names.

Value

A covariance matrix in the fit's free-parameter order.

<code>version</code>	<i>Native library version</i>
----------------------	-------------------------------

Description

Reports the version of the compiled magmaan library.

Usage

`version()`

Value

A version string.

Index

association_ml_global_test, 4
association_ml_ij, 5
association_ml_modification_indices,
6
association_ml_nested_test
(*association_ml_global_test*),
4
association_ml_score_tests
(*association_ml_modification_indices*),
6
augment_model_spec (*prepare_model*),
89

bounds_loading, 7
bounds_pos_var, 7
bounds_standard, 8
bounds_variance, 8
bounds_wide, 9

calibrate_quadratic (*scores*), 100
case_rerun, 9
case_rerun(), 22, 29
composite_weights, 10
compute_defined, 10
compute_defined(), 72
continuous_invariance, 11
convention_fit_measures, 12
convention_inference, 12
convention_inference(), 71
convention_nested
(*convention_inference*), 12
convergence_sim, 13
convergence_sim_catalog, 14
convergence_sim_dejonckere_crossloading,
14
convergence_sim_dejonckere_msst, 15
convergence_sim_dejonckere_shrinkage,
15
convergence_sim_dejonckere_simple, 16
convergence_sim_ludtke_cfa, 17

df_to_data, 17
df_to_fcsem_data, 18
df_to_fiml_data, 19

eqs_model, 19
est_change, 20
est_change(), 21, 22
est_change_approx, 21
est_change_raw, 22
est_change_raw(), 23
est_change_raw_approx, 23
est_change_raw_approx(), 21, 22
estimate (*prepare_model*), 89

factor_score_precision, 24
factor_scores, 24
fcsem_fit_measures, 25
fcsem_model_spec, 25
fcsem_standard_errors, 26
fcsem_standardized_rows, 26
fit_index_intervals_misspec, 27
fit_measures, 28
fit_measures(), 29
fit_measures_change, 28
fit_measures_misspec, 29
fit_measures_misspec(), 31
fit_measures_misspec_mixed_ordinal,
30
fit_measures_noniterative, 31
fit_ml_fcsem, 32
fit_model, 32
fit_model(), 12, 36, 71, 72, 93
fit_noniterative_cfa, 34
fit_noniterative_cfa(), 31, 75–83
fit_noniterative_cfa_metric, 36
fit_noniterative_cfa_restricted, 37
fit_noniterative_cfa_restricted(),
80
fit_twolevel, 39
fmgn_nested, 40

- fmg_nested_mixed_ordinal
 (*fmg_nested_ordinal*), 41
- fmg_nested_ordinal, 41
- fmg_pvalues, 42
- fmg_tests, 43
- fmg_tests(), 31, 42
- fmg_tests_mixed_ordinal
 (*fmg_tests_ordinal*), 44
- fmg_tests_ordinal, 44
- fmg_tests_ordinal(), 41, 42
- frontier_fit_fiml_multiinfo, 45
- frontier_fit_fiml_psd, 46
- frontier_fit_gls_psd, 46
- frontier_fit_mixed_ordinal_psd, 47
- frontier_fit_ml2s_psd, 51
- frontier_fit_ml_multiinfo, 48
- frontier_fit_ml_psd, 49
- frontier_fit_ml_psd_fallback, 50
- frontier_fit_ordinal_psd, 52
- frontier_fit_sphere, 53
- frontier_fit_uls_psd, 54
- frontier_fit_wls_psd, 55
- frontier_newton_accuracy, 56
- frontier_reidentify, 57

- global_score_flip_test, 57
- guttman_h, 59

- infer_build_u_factor
 (*lab_inference_primitives*),
 60
- infer_build_u_factor_fit
 (*lab_inference_primitives*),
 60
- infer_build_u_factor_pairwise
 (*lab_inference_primitives*),
 60
- infer_build_u_factor_parts
 (*lab_inference_primitives*),
 60
- infer_continuous_ls_profile_lrt
 (*lab_inference_primitives*),
 60
- infer_continuous_ls_robust
 (*lab_inference_primitives*),
 60
- infer_ml_profile_lrt
 (*lab_inference_primitives*),
 60

- infer_ordinal_robust_ij
 (*robust_ordinal_ij*), 95
- inference_covariance
 (*prepare_inference_data*), 87
- inference_information (scores), 100
- inference_quadratic
 (*prepare_inference_data*), 87
- inference_reuse
 (*prepare_inference_data*), 87
- inference_rows, 60

- lab_inference_primitives, 60
- lav_residuals, 62

- magmaan_core, 63
- magmaan_fcsem, 63
- mahalanobis_rerun, 64
- measures_reliability_cov
 (*lab_inference_primitives*),
 60
- measures_reliability_omega_multidim
 (*lab_inference_primitives*),
 60
- mi_screen, 64
- model_spec, 65
- modification_indices, 66
- modification_indices_lrt, 67
- modification_indices_robust, 67
- mplus_data, 70
- mplus_data(), 72
- mplus_model, 71
- mplus_model(), 70
- mplus_wlsmv_invariance, 73

- nested_score_test, 74
- nested_score_test(), 98
- nestedTest (*robust_nested_lrt*), 93
- noniterative_cfa_constrained, 75
- noniterative_cfa_constrained(), 36
- noniterative_cfa_difference_test, 76
- noniterative_cfa_grouped_inference,
 77
- noniterative_cfa_inference, 78
- noniterative_cfa_inference(), 77
- noniterative_cfa_modification_indices,
 79
- noniterative_cfa_pseudo_lrt, 80
- noniterative_cfa_scalar_invariance,
 81

noniterative_cfa_se, 82
 noniterative_cfa_wald, 82

 parameter_covariance (*scores*), 100
 parameter_table, 83
 policy_fit_measures, 28, 84
 policy_inference, 85
 policy_mi_refit, 85
 policy_modification_indices, 86, 86
 policy_nested, 86, 87
 prepare_data (*prepare_model*), 89
 prepare_hypothesis
 (*prepare_inference_data*), 87
 prepare_inference (*scores*), 100
 prepare_inference_data, 87
 prepare_model, 89
 prepare_weight (*prepare_model*), 89
 project_scores (*scores*), 100

 quadratic_reference (*scores*), 100

 refit_from_null, 92
 resample_scores (*scores*), 100
 robust_nested_lrt, 93
 robust_nested_lrt(), 40
 robust_ordinal_ij, 95

 sam, 96
 score_components (*scores*), 100
 score_components_from_matrices
 (*scores*), 100
 score_flip_test, 97
 score_flip_test(), 57, 74
 score_quadratic (*scores*), 100
 score_sandwich (*scores*), 100
 score_spectrum (*scores*), 100
 score_tests, 98
 score_tests_lrt, 99
 score_tests_robust
 (*modification_indices_robust*),
 67
 scores, 100
 sim_model_batch, 102
 sim_model_calibrate, 103
 sim_model_calibrate(), 104
 sim_model_draw, 103
 sim_ordcorr_batch, 104
 sim_ordcorr_calibrate, 105
 sim_ordcorr_calibrate(), 106, 109

 sim_ordcorr_draw, 106
 sim_ordcorr_mg_batch, 107
 sim_ordcorr_mg_calibrate, 108
 sim_ordcorr_mg_draw, 109
 sim_ordcorr_mg_summary_calibrate, 109
 sim_ordcorr_summary_calibrate, 110
 standardized, 111
 standardized_rows, 111
 structural_identification, 112

 vcov.magmaan_fit, 112
 version, 114

 wald_test (*scores*), 100