# Package: lavDiag (via r-universe)

May 16, 2026

**Title** Latent Variable Models Diagnostics

**Version** 0.1.0

**Description** Diagnostics and visualization tools for latent variable models fitted with 'lavaan' (Rosseel, 2012 <doi:10.18637/jss.v048.i02>). The package provides fast, parallel-safe factor-score prediction (lavPredict_parallel()), data augmentation with model predictions, residuals, delta-method standard errors and confidence intervals (augment()), and model-based latent grids for continuous, ordinal, or mixed indicators (prepare()). It offers item-level empirical versus model curve comparison using generalized additive models for both continuous and ordinal indicators (item_data(), item_plot()) via 'mgcv' (Wood, 2017, ISBN:9781498728331), residual diagnostics including residual correlation tables and plots (resid_cor(), resid_corrplot()) using 'corrplot' (Wei and Simko, 2021 <https://github.com/taiyun/corrplot>), and Q–Q checks of residual z-statistics (resid_qq()), optionally with non-overlapping labels from 'ggrepel' (Slowikowski, 2024 <https://CRAN.R-project.org/package=ggrepel>). Heavy computations are parallelized via 'future'/'furrr' (Bengtsson, 2021 <doi:10.32614/RJ-2021-048>; Vaughan and Dancho, 2018 <https://CRAN.R-project.org/package=furrr>). Methods build on established literature and packages listed above.

**License** MIT + file LICENSE

**URL** https://github.com/reckak/lavDiag

**BugReports** https://github.com/reckak/lavDiag/issues

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1.0)

**Imports** corrplot (>= 0.90), dplyr, furrr, future, future.apply, ggplot2, ggrepel, igraph, lavaan, mgcv, purrr, rlang, stringr, tibble, tidyselect, tidyr, vctrs, visNetwork, withr

**Suggests** spelling, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Config/pak/sysreqs** cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev

**Repository** https://reckak.r-universe.dev

**Date/Publication** 2026-01-20 17:40:45 UTC

**RemoteUrl** https://github.com/reckak/lavdiag

**RemoteRef** HEAD

**RemoteSha** fbef85c5c225abf982644b20dacb711d1b356e3b

# Contents



---

`augment` *Augment SEM data with predictions, residuals, SEs/CIs, and ordinal extras*

---

## Description

User-facing wrapper that augments a fitted **lavaan** model with: predicted observed values (*yhat*), residuals (*obs - yhat*), delta-method standard errors and confidence intervals for predictions, and—when the model includes ordinal indicators—latent linear predictors (*y**) and per-category probabilities. Works for continuous-only, ordinal-only, and mixed models by internally routing to specialized helpers for each measurement type.

## Usage

```
augment(
  fit,
  data = NULL,
  info = NULL,
  yhat = TRUE,
  resid = TRUE,
  ci = TRUE,
  level = 0.95,
  se_yhat = TRUE,
  ystar = TRUE,
  pr = TRUE,
  se_fs = TRUE,
  vcov_type = NULL,
  col_layout = c("by_type", "by_item"),
  prefix_ystar = ".ystar_",
  prefix_yhat = ".yhat_",
  prefix_pr = ".pr_",
  prefix_ci = c(".yhat_lwr_", ".yhat_upr_"),
  prefix_resid = ".resid_",
  prefix_se_fs = ".se_",
  prefix_se_yhat = ".se_yhat_",
  sep = "__"
)
```

## Arguments

| | |
|---|---|
| fit | A fitted lavaan model object. |
| data | Optional factor-score output to reuse. Either a data.frame (single-group) or a list of data.frames (per group) as returned by lavPredict_parallel(). If NULL, it is computed once and reused internally. |
| info | Optional model_info() list; if NULL, it is computed. |
| yhat | Logical; include predicted observed values. Default TRUE. |
| resid | Logical; include residuals (obs - yhat). Default TRUE. |
| ci | Logical; include delta-method confidence intervals for yhat. Default TRUE. |
| level | Confidence level for ci. Default 0.95. |
| se_yhat | Logical; include delta-method standard errors of yhat. Default TRUE. |
| ystar | Logical; for ordinal items, include latent linear predictors $y^*$. Default TRUE. |
| pr | Logical; for ordinal items, include per-category probabilities. Default TRUE. |
| se_fs | Logical; request factor-score SEs from lavPredict_parallel() for continuous-only models. Ignored when the model contains any ordinal indicators. Default TRUE. |
| vcov_type | Optional vcov type passed to the continuous branch when computing robust SEs/CIs for predictions (e.g., "sandwich"). |

| | |
|---|---|
| col_layout | Column layout for the augmented output; either "by_type" (observed, y*, yhat, CI lower/upper, se_yhat, residuals, probabilities) or "by_item" (per-item blocks). Default c("by_type","by_item") with match.arg() semantics. |
| prefix_ystar | Character prefix for latent linear predictor columns (ordinal), e.g., ".ystar_". Default ".ystar_". |
| prefix_yhat | Character prefix for predicted observed values, e.g., ".yhat_". Default ".yhat_". |
| prefix_pr | Character prefix for ordinal category-probability columns, e.g., ".pr_". Default ".pr_". |
| prefix_ci | Length-2 character vector with lower/upper prefixes for prediction intervals, e.g., c(".yhat_lwr_", ".yhat_upr_"). Default c(".yhat_lwr_", ".yhat_upr_"). |
| prefix_resid | Character prefix for residual columns, e.g., ".resid_". Default ".resid_". |
| prefix_se_fs | Character prefix for factor-score SE columns (continuous-only), e.g., ".se_". Default ".se_". |
| prefix_se_yhat | Character prefix for prediction SE columns, e.g., ".se_yhat_". Default ".se_yhat_". |
| sep | Separator used in probability column names between category and item, e.g., "__" giving names like ".pr_2__Item1". Default "__". |

## Details

Internally, augment() delegates work to the internal functions .augment_continuous() and .augment_ordinal(), each optimized for their respective indicator type. The wrapper automatically detects whether the fitted model contains continuous, ordinal, or mixed indicators and merges outputs from both branches as needed.

The function reuses optional inputs:

- data: precomputed factor scores (and optional FS SEs) as returned by lavPredict_parallel() to avoid duplicate work.
- info: model metadata from model_info().

Column naming is controlled by prefix arguments. For ordinal probabilities, names follow the pattern <prefix_pr><category><sep><item>, e.g. ".pr_3__A1" for category 3 of item A1 when sep = "__". The final column order can be arranged either "by type" (all observed, then all y*, all yhat, CIs, SEs, residuals, then probabilities) or "by item" (grouping each item's block together) via col_layout.

## Value

A tibble-like data.frame. Columns appear in the following order and with the following rules:

- **(i) Anchors first**: .rid, .gid, .group always lead the table.
- **(ii) Original lavPredict columns next**: observed indicators, factor scores, and (optionally) factor-score SEs returned by lavPredict_parallel() follow immediately after anchors.
- **(iii) FS SEs presence**: factor-score SE columns are present *only* when se_fs = TRUE *and* the model is continuous-only (i.e., contains no ordinal indicators).
- **(iv) Augmentations**: per measurement type, adds y* (for ordinal), yhat, CI lower/upper, se_yhat, residuals, and (for ordinal) per-category probabilities.

- **(v) Probability naming**: ordinal probability columns follow the pattern `<prefix_pr><cat><sep><item>` (e.g., `".pr_3__A1"`).

The relative order of the augmentation blocks (observed/y*/yhat/CIs/SEs/residuals/probabilities) is controlled by `col_layout`.

## See Also

Other lavDiag-augmenters: `item_data()`, `prepare()`

## Examples

```
# Continuous example
HS.model <- 'visual  =~ x1 + x2 + x3
             textual =~ x4 + x5 + x6
             speed   =~ x7 + x8 + x9'
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939,
                   meanstructure = TRUE)
augment(fit)


# --- Ordinal example (discretize by quantiles; 5 ordered categories) -------
ord_items <- paste0("x", 1:9)
HS_ord <- lavaan::HolzingerSwineford1939
for (v in ord_items) {
  q <- stats::quantile(HS_ord[[v]], probs = seq(0, 1, length.out = 6), na.rm = TRUE)
  q <- unique(q)  # guard against duplicate cut points
  HS_ord[[v]] <- as.ordered(cut(HS_ord[[v]], breaks = q, include.lowest = TRUE))
}

fit_ord <- lavaan::cfa(
  HS.model,
  data             = HS_ord,
  ordered          = ord_items,
  estimator        = "WLSMV",
  parameterization = "delta",
  meanstructure    = TRUE
)
augment(fit_ord)

# --- Mixed example (x1–x3 ordinal, others continuous) ----------------------
mix_ord <- c("x1","x2","x3")
HS_mix  <- lavaan::HolzingerSwineford1939
for (v in mix_ord) {
  q <- stats::quantile(HS_mix[[v]], probs = seq(0, 1, length.out = 6), na.rm = TRUE)
  q <- unique(q)
  HS_mix[[v]] <- as.ordered(cut(HS_mix[[v]], breaks = q, include.lowest = TRUE))
}

fit_mix <- lavaan::cfa(
  HS.model,
  data             = HS_mix,
```

```
  ordered          = mix_ord,
  estimator        = "WLSMV",
  parameterization = "delta",
  meanstructure    = TRUE
)
augment(fit_mix)
```

---

`hopper_plot` *Hopper plot of the largest residual correlations*

---

## Description

Draws a "hopper" plot of the top-`n_max` absolute residual correlations (Bentler type) computed by `resid_cor()` — either for a single group or per-group facets for multi-group models.

## Usage

```
hopper_plot(fit, title = NULL, n_max = 15, sep = "___")
```

## Arguments

| | |
|---|---|
| `fit` | A fitted `lavaan` object. |
| `title` | Optional plot title. |
| `n_max` | Number of variable pairs to show (per group when multi-group). |
| `sep` | Separator used by `.reorder_within()` for multi-group axis labeling. You usually don't need to change this. |

## Value

A `ggplot2` object.

## See Also

resid_cor, resid_corrplot

Other lavDiag-visualization: `item_plot()`

## Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939)
hopper_plot(fit, n_max = 10)
```

item_data *Compute Empirical and Model-Based Item Curves with Fit Diagnostics*

## Description

High-level wrapper that combines model-based (from `augment()`) and empirical (from GAM fits) item-level predictions for continuous and ordinal indicators. For each item, smooth GAM curves are estimated as functions of the latent variables on which the item loads, optionally in a multi-group setup.

The function returns both the augmented dataset containing model-based and empirical predictions (`original_data`), a table of item-level fit metrics (`metrics`), and optionally predicted empirical values for latent grids (`new_data`).

## Usage

```
item_data(
  fit,
  data = NULL,
  info = NULL,
  level = 0.95,
  fam_cont = mgcv::betar(link = "logit"),
  fam_ord = "ocat",
  gam_args_cont = list(method = "REML"),
  gam_args_ord = list(method = "REML", select = TRUE),
  plan = c("auto", "multisession", "multicore", "sequential", "cluster", "none"),
  workers = NULL,
  cluster = NULL,
  progress = FALSE,
  verbose = TRUE,
  store_fits = TRUE
)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted `lavaan` or `blavaan` model object. Must include latent variables. |
| `data` | Optional data frame with observed indicators used in the model. If `NULL`, data are taken from the fitted model. |
| `info` | Optional output of `model_info`. If `NULL`, it will be computed automatically. |
| `level` | Numeric; confidence level for prediction intervals (default = 0.95). |
| `fam_cont` | A `mgcv` family object for continuous indicators (default = `mgcv::betar(link = "logit")`). |
| `fam_ord` | Accepts either the string `"ocat"` or any valid `mgcv` family object for ordinal indicators. In mixed models, the function automatically selects the appropriate branch for each indicator type. |

| | |
|---|---|
| gam_args_cont | A named list of arguments passed to GAM fitting for continuous items. |
| gam_args_ord | A named list of arguments passed to GAM fitting for ordinal items. |
| plan | Parallelization backend; one of "auto", "multisession", "multicore", "sequential", "cluster", or "none". |
| workers | Optional integer; number of parallel workers to use. |
| cluster | Optional external cluster object (e.g., from parallel::makeCluster()). |
| progress | Logical; whether to display a progress bar (default = FALSE). |
| verbose | Logical; whether to print progress messages (default = TRUE). |
| store_fits | Logical; whether to store fitted GAM models and use them to compute predictions for latent grids via prepare() (default = TRUE). |

## Details

Internally, the function:

1. Verifies that the input lavaan/blavaan model converged and contains latent variables.
2. Extracts factor loadings and thresholds per group.
3. Runs parallelized GAM fitting (mgcv::gam) for each item and group, using either Gaussian, Beta, or ordinal (ocat) families.
4. Produces empirical item curves and agreement metrics between model-based and empirical predictions ($R^2$, RMSE, MAE, and penalized variants).
5. Optionally, predicts empirical curves for latent grids produced by prepare().

## Value

A list with three elements:

original_data A tibble containing the augmented original dataset with both model-based (m_est_yhat_*) and empirical (e_est_yhat_*) predictions for each item.

metrics A tibble summarizing item-level fit indices for each group and item:

- r2, rmse, mae: agreement between model and empirical fits.
- r2_pen, rmse_pen, mae_pen: penalized variants accounting for model complexity.
- c_m, c_e, k_eff: effective model and empirical complexity measures.

new_data Optional tibble of latent-grid predictions including empirical estimates (e_est_* etc.); present only when store_fits = TRUE.

## Parallelization

Parallel execution is handled via furrr::future_map() and controlled by plan, workers, and cluster. Default is "auto", which attempts to use an optimal backend based on the operating system.

### Agreement Metrics

- $$R^2 = \mathrm{cor}^2(m, e)$$
- $$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(m_i - e_i)^2}$$
- $$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|m_i - e_i|$$

### See Also

augment, prepare, model_info, gam, future_map

Other lavDiag-augmenters: augment(), prepare()

### Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939,
                   meanstructure = TRUE)
item_data(fit)
```

---

## item_plot *Plot model-implied vs empirical item curves by latent factor (single- or multi-group)*

---

### Description

Given the output list of item_data(), draw smooth curves of the model-implied relation and the empirical GAM-based relation between a chosen latent factor (x-axis) and all items that load on it (y-axis). Works for continuous, ordinal, and mixed models; supports single- and multi-group fits.

### Usage

```
item_plot(
  x,
  latent,
  items = NULL,
  show_model = TRUE,
```

```
  show_empirical = TRUE,
  show_points = TRUE,
  show_metrics = TRUE,
  metrics_pad = 0.05,
  point_size = 1.2,
  line_width = 0.9,
  color_points = "black",
  color_model = "blue",
  color_empirical = "red",
  alpha_points = 0.05,
  alpha_ribbon = 0.2,
  jitter_sd = 0.05,
  jitter_seed = NULL,
  sample_frac = 1,
  point_shape = 16,
  ribbons = TRUE,
  penalized = FALSE,
  facet = c("wrap", "grid"),
  ncol = NULL,
  nrow = NULL,
  scales = c("fixed", "free_y", "free", "free_x"),
  sort = c("none", "r2", "r2_pen", "rmse", "rmse_pen", "mae", "mae_pen"),
  sort_dir = c("auto", "asc", "desc")
)
```

## Arguments

| | |
|---|---|
| `x` | A list returned by `item_data()`. |
| `latent` | Character scalar: latent factor ID shown on x-axis. |
| `items` | Optional character vector of item names to include; inferred if `NULL`. |
| `show_model`, `show_empirical`, `show_points` | Logical toggles to draw the model curve, the empirical curve, and raw datapoints (all `TRUE` by default). |
| `show_metrics` | Logical: print per-item fit metrics inside each facet (default `TRUE`). |
| `metrics_pad` | Additional top padding for facets when `show_metrics = TRUE`. Controls the gap between metric labels and the plotting area (default `0.05`). |
| `point_size` | Numeric size of points (defaults chosen for legibility). |
| `line_width` | Numeric linewidth for model/empirical curves (defaults chosen for legibility). |
| `color_points`, `color_model`, `color_empirical` | Colors for points, model curve/ribbon, empirical curve/ribbon (defaults chosen for legibility). |
| `alpha_points`, `alpha_ribbon` | Alphas for points and ribbons (defaults chosen for legibility). |
| `jitter_sd` | Vertical jitter SD (on the data scale); normally distributed. The applied jitter is truncated to ±3·SD to avoid extreme outliers. |
| `jitter_seed` | Optional integer seed for deterministic vertical jitter. |

| | |
|---|---|
| sample_frac | Optional fraction in (0,1] to thin raw datapoints before plotting (default 1 = no thinning). |
| point_shape | Integer/character ggplot2 shape for points (choose a fast filled shape, default 16). |
| ribbons | Logical: draw CI ribbons when available (default TRUE). |
| penalized | Logical: use penalized metrics in facet captions (default FALSE). |
| facet | One of "wrap", "grid". Multi-group inputs always use grid. |
| ncol, nrow | Optional layout hints used when facet = "wrap". |
| scales | Facet scales (passed to ggplot2 facets). Default "fixed". |
| sort | Item ordering by a metric; one of "none", "r2", "r2_pen", "rmse", "rmse_pen", "mae", "mae_pen" (default "none"). For multi-group, the metric is averaged across groups. |
| sort_dir | Direction of sort if sort != "none"; one of "auto", "asc", "desc". auto sorts $R^2$-like metrics descending and error metrics ascending. |

## Value

A ggplot2 object (single-group) or a faceted ggplot2 object (multi-group). The plot is returned (and printed if not assigned) and can be further modified with +.

## See Also

item_data, augment, prepare

Other lavDiag-visualization: hopper_plot()

## Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939,
                   meanstructure = TRUE)
idata <- item_data(fit)
item_plot(idata, latent = 'visual')

# Multi-group example (facet grid, sort by R2)
fit_mg <- lavaan::cfa(HS.model,
                      data = lavaan::HolzingerSwineford1939,
                      group = 'school', meanstructure = TRUE)
idata_mg <- item_data(fit_mg)
item_plot(idata_mg, latent = 'visual', facet = 'grid', sort = 'r2')
```

`lavPredict_parallel` *Fast & robust parallel wrapper for lavaan::lavPredict()*

## Description

Parallel, ordinal-aware, and mixed-safe implementation of `lavaan::lavPredict()`.

## Usage

```
lavPredict_parallel(
  fit,
  method = "ml",
  correct_extremes = TRUE,
  extreme_rule = c("auto", "abs", "z_by_se", "mad"),
  extreme_by = c("auto", "group", "global"),
  extreme_k = 4,
  extreme_eps = 1e-08,
  fallback_method = c("EMB", "EBM"),
  flag_column = FALSE,
  diagnostics = FALSE,
  workers = NULL,
  plan = c("auto", "multisession", "multicore", "sequential"),
  chunk_size = NULL,
  return_type = c("list", "data"),
  progress = FALSE,
  se = FALSE,
  prefix_se_fs = ".se_",
  distinct = c("never", "auto", "always"),
  distinct_threshold = 50000,
  ...
)
```

## Arguments

| | |
|---|---|
| `fit` | lavaan model object. |
| `method` | Character; estimation method passed to `lavaan::lavPredict()` (e.g., `"ml"`, `"regression"`, `"EBM"`, `"EMB"`). |
| `correct_extremes` | Logical; if `TRUE`, re-score only flagged rows via `fallback_method`. |
| `extreme_rule` | One of `"auto"`, `"abs"`, `"z_by_se"`, `"mad"`. See details above. |
| `extreme_by` | One of `"auto"`, `"group"`, `"global"`. If `"group"`, thresholds are computed within groups. |
| `extreme_k` | Numeric; rule-dependent threshold (default = 3.5). |
| `extreme_eps` | Small numeric to guard division by zero in z-scores (default = 1e-8). |

`fallback_method`
: Character; fallback method(s) for corrected rows. Default tries `"EMB"` first, then `"EBM"` if that fails.

`flag_column`
: Logical; if `TRUE`, add column `.fs_corrected` marking corrected rows.

`diagnostics`
: Logical; if `TRUE`, attach diagnostic columns and attribute `fs_n_corrected`.

`workers`
: Integer; number of parallel workers (default = `max(1, parallel::detectCores() - 1)`).

`plan`
: One of `"auto"`, `"multisession"`, `"multicore"`, `"sequential"`. `"auto"` selects `"multisession"` cross-platform.

`chunk_size`
: Optional integer; number of rows per chunk (default computed adaptively).

`return_type`
: `"list"` or `"data"`; determines multi-group return format.

`progress`
: Logical; show `furrr` progress bar.

`se`
: Logical; if `TRUE` and model is continuous-only, attach standard-error columns for factor scores.

`prefix_se_fs`
: Character; prefix for SE columns (default = `".se_"`).

`distinct`
: One of `"never"`, `"auto"`, `"always"`; controls use of `distinct()` to shrink duplicated rows.

`distinct_threshold`
: Numeric; when `distinct = "auto"`, apply `distinct()` only if `nrow(data) >= distinct_threshold` (default = 5e4). Ignored for mixed models.

`...`
: Additional arguments passed to `lavaan::lavPredict()`.

## Details

**Model-type behavior:**

- **Purely ordinal models:** Optionally reduces duplicated work via `dplyr::distinct()` on full rows. Prepends "all-categories" dummy rows per group to stabilize chunk predictions.
- **Mixed models (ordinal + continuous):** No deduplication (predictions depend on continuous values). Additionally prepends "variance-insurance" dummy rows to ensure non-zero variance of all continuous variables within each group.

**Robust extreme-score handling:**

If `correct_extremes = TRUE`, only flagged rows are re-scored using a fallback method. The flagging rule is controlled by `extreme_rule`:

- `"auto"` (default): per-latent mixed metric — uses `"z_by_se"` where SE columns exist for that latent, otherwise robust `"mad"`. A row is flagged if **any** latent exceeds the threshold; the maximum across latents is used.
- `"z_by_se"`: flags rows where `abs(FS) / pmax(SE, eps) > extreme_k` for any latent.
- `"mad"`: robust Z: `abs(FS - median)/(1.4826 × MAD) > extreme_k`.
- `"abs"`: simple absolute threshold: `abs(FS) > extreme_k`.

The re-scoring uses the first `fallback_method` supplied (default `"EMB"`), and then automatically retries the other (e.g. `"EBM"`) if needed.

If `flag_column = TRUE`, a logical column `.fs_corrected` marks corrected rows. If `diagnostics = TRUE`, columns `.fs_rule` and `.fs_metric` are attached and an attribute `fs_n_corrected` is added. The internal `"mixed"` rule is always reported as `"auto"` for user clarity.

## Value

A tibble (single-group) or a list/tibble (multi-group, depending on `return_type`), containing predicted factor scores and optionally SEs and diagnostics.

## Examples

```
# Convert selected indicators to ordinal
ord_items <- paste0("x", 1:9)
HS_ord <- lavaan::HolzingerSwineford1939
for (v in ord_items) {
  q <- stats::quantile(HS_ord[[v]], probs = seq(0, 1, length.out = 6), na.rm = TRUE)
  q <- unique(q)  # guard against duplicate cut points
  HS_ord[[v]] <- as.ordered(cut(HS_ord[[v]], breaks = q, include.lowest = TRUE))
}

HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

# Fit ordinal CFA model
fit_ord <- lavaan::cfa(
  HS.model,
  data             = HS_ord,
  ordered          = ord_items,
  estimator        = "WLSMV",
  parameterization = "delta",
  meanstructure    = TRUE
)

# Parallel prediction with automatic extreme-score handling
lavPredict_parallel(fit_ord, correct_extremes = TRUE)
```

---

`model_info` *Extract compact, robust meta-information about a lavaan fit*

---

## Description

Lightweight helper that queries a fitted **lavaan** object for commonly needed model metadata (grouping, variables, estimator, parameterization, categorical/multilevel flags, etc.). All lookups are wrapped in `tryCatch()`, so the function returns informative `NA`/`NULL` values instead of failing when particular slots are unavailable.

## Usage

```
model_info(fit)
```

## Arguments

| | |
|---|---|
| fit | A fitted lavaan model object. |

## Details

The function relies on stable lavInspect / lavNames queries with minimal post-processing:

- Grouping: number of groups, labels, grouping variable (if any), and per-group sample sizes.
- Variables: observed (ov) and latent (lv) in model order; observed split into ordinal vs. continuous using lavNames(type = "ov.ord").
- Estimator and parameterization: taken from lavInspect(fit, "options").
- Multilevel summary: coarse flags derived from "nlevels" / "cluster" and related quantities (clusters, average cluster size).

All fields are safe to access: if a query is not applicable (e.g., single-group model has no "group"), the corresponding element is set to NA, NULL, or a sensible default.

## Value

A named list with the following elements:

converged Logical or NA; model convergence flag.

has_meanstructure Logical or NA; whether a mean structure was estimated.

estimator Character or NA; e.g., "ML", "WLSMV", "Bayes".

parameterization Character or NA; e.g., "delta", "theta".

is_single_group Logical; TRUE for single-group models.

n_groups Integer or NA; number of groups.

group_var Character or NULL; name of the grouping variable.

group_labels Character vector or NULL; group labels in model order.

n_obs Per-group sample sizes (vector/list) or NULL.

observed_variables Character; observed indicators (ov) in model order.

latent_variables Character; latent variables (lv) in model order.

ov_ordinal Character; subset of ordinal observed variables.

ov_continuous Character; observed variables not in ov_ordinal.

is_categorical Logical or NA; lavaan-level categorical flag.

is_multilevel Logical; TRUE if a multilevel structure is detected.

n_levels Integer or NA; number of levels.

cluster_var Character vector or NULL; clustering variable(s), if any.

n_clusters Integer or NA; number of clusters (if available).

average_cluster_size Numeric or NA; average cluster size (if available).

## See Also

prepare, augment, item_data; lavInspect, lavNames

## Examples

```
HS.model <- 'visual  =~ x1 + x2 + x3
             textual =~ x4 + x5 + x6
             speed   =~ x7 + x8 + x9'
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939,
                   group = 'school')
model_info(fit)
```

---

`parameter_estimates` *Parameter estimates from lavaan with a unified schema*

---

## Description

Extracts raw or standardized coefficients from a fitted lavaan model, always ensuring a `group` column is present (set to 1 for single-group models). Internally, the function relies on `lavaan::parameterEstimates()` for raw estimates and `lavaan::standardizedSolution()` for standardized coefficients.

## Usage

```
parameter_estimates(
  fit,
  level = 0.95,
  standardized = "none",
  include_r2 = TRUE
)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted `lavaan` object. |
| `level` | Confidence level for intervals (default `0.95`). |
| `standardized` | Either a logical (FALSE/TRUE) or one of `c("none", "std.all", "std.lv", "std.nox")`. If `TRUE`, it is treated as `"std.all"`. |
| `include_r2` | Logical; include R-squared rows (only when `standardized = "none"`), default `TRUE`. |

## Details

This wrapper harmonizes the output structure between raw and standardized estimates, renames standardized columns to a unified schema (e.g., `est` instead of `est.std`), and ensures that a `group` column is always included. When `include_r2 = TRUE`, $R^2$ values are appended for each endogenous variable if available in the model.

## Value

A `data.frame` in the style of `lavaan::parameterEstimates()`. If standardized output is requested, the estimate is in column `est` (renamed from `est.std` for consistency).

## Examples

```
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939)

# Raw estimates with R2
pe  <- parameter_estimates(fit)

# Standardized (std.all)
pes <- parameter_estimates(fit, standardized = TRUE)

# Standardized (std.lv)
pes_lv <- parameter_estimates(fit, standardized = "std.lv")
```

---

# plot_cfa *Interactive CFA/SEM diagram via visNetwork*

---

## Description

Builds interactive confirmatory factor analysis (CFA) or structural equation model (SEM) diagram(s) for a selected type of estimates:

- `"none"` – raw (unstandardized) estimates,
- `"std.all"` – fully standardized estimates,
- `"std.lv"` – standardized by latent variances.

For multi-group models, one interactive widget is produced per group; for single-group models, a single widget is returned.

## Usage

```
plot_cfa(fit, standardized = "none", include_r2 = TRUE)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted `lavaan` or `blavaan` model object. |
| `standardized` | Either a logical scalar or one of `c("none", "std.all", "std.lv")`. If `TRUE`, equivalent to `"std.all"`; default is `"none"`. |
| `include_r2` | Logical; if `TRUE`, include $R^2$ values for dependent variables in node tooltips (computed from raw estimates via `parameterEstimates(rsquare = TRUE)`). Default `TRUE`. |

## Details

The diagram visualizes all latent and observed variables and connects them according to the fitted model. Edge thickness and opacity are scaled by the squared standardized estimate. Tooltips on edges display parameter estimates, confidence intervals, and significance tests. Tooltips on nodes summarize residual variances, intercepts, thresholds (if categorical), and optionally $R^2$ values.

The function relies on stable summaries from `model_info()` and `parameter_estimates()`, and uses internal helpers (e.g., `.fnum()`, `.ci()`) for compact formatting.

## Value

A list of **visNetwork** `htmlwidgets`:

- length 1 for single-group models,
- one element per group for multi-group models.

## See Also

`model_info`, `parameter_estimates`,

## Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(
  HS.model,
  data = lavaan::HolzingerSwineford1939
)
plot_cfa(fit, standardized = "std.all")
```

---

# prepare *Prepare smooth latent grids + model-based item curves (continuous, ordinal, or mixed)*

---

## Description

Unified, robust wrapper that constructs synthetic latent-score grids and model-based item curves for fitted `lavaan`/`blavaan` models. It tries both internal branches and returns whichever applies:

- **Continuous branch** (for continuous indicators) and
- **Ordinal branch** (for ordinal indicators).

If both succeed (mixed models), their outputs are joined on shared ID columns. The function is designed to avoid manual handling of factor-score data or group columns; that logic is delegated to sub-functions.

## Usage

```
prepare(
  fit,
  data = NULL,
  info = NULL,
  plan = c("auto", "none", "multisession", "multicore", "sequential", "cluster"),
  workers = NULL,
  cluster = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| fit | A fitted lavaan/blavaan object. |
| data | Optional factor-score table to reuse (either a single data frame or a per-group list) as typically returned by lavPredict_parallel(). If NULL, factor scores are obtained on demand by the sub-functions. |
| info | Optional list from model_info(fit). If NULL, it is computed once and forwarded to both branches. |
| plan | Parallelization backend for the **ordinal** branch; one of c("auto", "none", "multisession", "multicore", "sequential", "cluster"). The continuous branch is fast and does not use futures. |
| workers | Optional integer; number of workers used by future backends (ignored unless plan uses a parallel backend). Forwarded to the ordinal branch. |
| cluster | Optional external cluster object created by parallel::makeCluster(); only used when plan = "cluster". If supplied with a different plan, it is ignored. |
| ... | Additional arguments passed unchanged to both sub-functions (e.g., level, vcov_type, length.out, other_latents, latent_var_as_factor, se, se_summary). |

## Details

### Routing

- Calls internal functions .prepare_continuous() and .prepare_ordinal() for continuous and ordinal items, respectively.
- Reuses info (from model_info()) and data (from lavPredict_parallel()) when provided, so each is computed at most once.

### Merging

- If only one branch applies, returns that result.
- If both apply, performs a dplyr::full_join() using shared keys among c(".rid", ".gid", ".group", ".latent_var") when available, otherwise on the intersection of column names.
- To prevent duplicated factor-score columns (e.g., A.x, E.y), the latent columns are kept from the **continuous** branch and dropped from the ordinal branch before the join.

### Column semantics

- Leading ID columns: `.rid`, `.gid`, `.group`, `.latent_var`.
- Latent columns follow directly after `.latent_var`.
- Continuous/ordinal model-based outputs use the common `m_est_*`, `m_lwr_*`, `m_upr_*` naming convention.

## Value

A tibble. For mixed models, it is the full join of the two branches using shared ID columns (see **Merging**). For single-type models, it is the single applicable branch.

## Parallelization

- The function may set a safe fallback to `sequential` when executed inside a worker to prevent nested futures. External clusters are respected only with `plan = "cluster"`. `workers` is a hint for future backends and is ignored otherwise.

## Errors and warnings

- If neither branch succeeds, the function aborts with a diagnostic message.
- When continuous and ordinal outputs have no shared columns, a row bind is returned with a `.source` column and a warning is issued.

## See Also

Internal helpers .prepare_continuous() and .prepare_ordinal() (not exported); see also `model_info()` and `lavPredict_parallel()`.

Other lavDiag-augmenters: `augment()`, `item_data()`

## Examples

```
# --- Continuous example ---------------------------------------------------
HS.model <- 'visual  =~ x1 + x2 + x3
             textual =~ x4 + x5 + x6
             speed   =~ x7 + x8 + x9'
fit <- lavaan::cfa(HS.model,
                   data = lavaan::HolzingerSwineford1939,
                   meanstructure = TRUE)
prepare(fit)


# --- Ordinal example (discretize by quantiles; 5 ordered categories) -----
ord_items <- paste0("x", 1:9)
HS_ord <- lavaan::HolzingerSwineford1939
for (v in ord_items) {
  q <- stats::quantile(HS_ord[[v]], probs = seq(0, 1, length.out = 6), na.rm = TRUE)
  q <- unique(q)  # guard against duplicate cut points
  HS_ord[[v]] <- as.ordered(cut(HS_ord[[v]], breaks = q, include.lowest = TRUE))
}

fit_ord <- lavaan::cfa(
```

```
  HS.model,
  data             = HS_ord,
  ordered          = ord_items,
  estimator        = "WLSMV",
  parameterization = "delta",
  meanstructure    = TRUE
)
prepare(fit_ord)

# --- Mixed example (x1–x3 ordinal, others continuous) --------------------
mix_ord <- c("x1","x2","x3")
HS_mix  <- lavaan::HolzingerSwineford1939
for (v in mix_ord) {
  q <- stats::quantile(HS_mix[[v]], probs = seq(0, 1, length.out = 6), na.rm = TRUE)
  q <- unique(q)
  HS_mix[[v]] <- as.ordered(cut(HS_mix[[v]], breaks = q, include.lowest = TRUE))
}

fit_mix <- lavaan::cfa(
  HS.model,
  data             = HS_mix,
  ordered          = mix_ord,
  estimator        = "WLSMV",
  parameterization = "delta",
  meanstructure    = TRUE
)
prepare(fit_mix)
```

---

`resid_cor` *Residual correlations (Bentler or other types) as a tidy tibble*

---

## Description

Creates a tidy tibble of residual **correlations** from a fitted `lavaan` model, including standard errors and z-statistics when available. Supports single- and multi-group models and allows selection of the correlation type (e.g., Bentler, Pearson, or residual covariance-based).

## Usage

```
resid_cor(
  fit,
  type = c("cor.bentler", "cor", "cor.sample", "cov", "cov.sample")
)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted `lavaan` object. |
| `type` | Character; which type of residual correlation to extract. One of `c("cor.bentler", "cor", "cor.sample", "cov", "cov.sample")`. Default is `"cor.bentler"`. |

## Details

Internally uses `lavaan::lavResiduals(type = type, se = TRUE)`. For multi-group models, a `group` column is added (using `lavaan::lavInspect(fit, "group.label")` when available).

**Duplicate removal & stable ordering**

- Residual correlations are first obtained as (group-wise) symmetric matrices.
- Only the *upper triangle without the diagonal* is kept, using a logic equivalent to `mat[upper.tri(mat, diag = FALSE)]`.
- Variable pairs are created via `v1 <- pmin(i, j)`, `v2 <- pmax(i, j)` so that each pair appears once regardless of original order.
- A human-readable `pair` label is created as `paste0(v1, "-", v2)`.
- The result is sorted stably by `group` (if present) and `pair` for reproducible outputs across sessions.

## Value

A tibble with columns:

- `v1`, `v2` – variable names in the pair
- `pair` – canonical pair label `"v1-v2"` with alphabetical ordering via `pmin/pmax`
- `cor` – residual correlation
- `abs_cor` – absolute value of `cor`
- `se` – standard error (if available from lavaan)
- `z` – z-statistic (if available)
- `group` – group label (multi-group models only)

## See Also

`resid_corrplot`, `hopper_plot`, `resid_qq`

## Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(HS.model, data = lavaan::HolzingerSwineford1939)
resid_cor(fit)
resid_cor(fit, type = "cor")  # standard residual correlations

# Multi-group example (group by school)
fit_mg <- lavaan::cfa(
  HS.model,
  data  = lavaan::HolzingerSwineford1939,
  group = "school"
)
```

```
rc <- resid_cor(fit_mg, type = "cor.bentler")
head(rc)
```

---

`resid_corrplot` *Corrplot of residual correlations (configurable type)*

---

## Description

Draw corrplot(s) of residual correlations from a fitted **lavaan** model. The `type` argument allows selecting which residual correlation or covariance metric to visualize (e.g., Bentler, Pearson, or sample-based). For multi-group models you can harmonize the color scale across groups via `common_scale = TRUE`. Produces base plots.

## Usage

```
resid_corrplot(
  fit,
  type = c("cor.bentler", "cor", "cor.sample", "cov", "cov.sample"),
  order = c("original", "AOE", "FPC", "hclust", "alphabet"),
  hclust.method = c("complete", "ward", "ward.D", "ward.D2", "single", "average",
    "mcquitty", "median", "centroid"),
  common_scale = TRUE,
  title_prefix = NULL,
  record = FALSE
)
```

## Arguments

| | |
|---|---|
| `fit` | A fitted `lavaan` object. |
| `type` | Character; which type of residual correlation/covariance to plot. One of `c("cor.bentler", "cor", "cor.sample", "cov", "cov.sample")`. Default `"cor.bentler"`. |
| `order` | One of `c("original","AOE","FPC","hclust","alphabet")` controlling the variable ordering in the plot. |
| `hclust.method` | One of `c("complete","ward","ward.D","ward.D2","single","average","mcquitty","median",` |
| `common_scale` | Logical; use a common symmetric color range across groups? Default `TRUE`. |
| `title_prefix` | Optional character prefix for multi-group plot titles. |
| `record` | Logical; if `TRUE`, return recorded plot(s) via `recordPlot()`. Default `FALSE`. |

## Details

Uses corrplot::corrplot.mixed() for rendering. When common_scale = TRUE, the color legend is harmonized across groups by taking a common symmetric range (-L, L) with $L = max(|values|)$ across all groups; otherwise each group panel uses its own range. The is.corr flag is set automatically based on type (TRUE when type starts with "cor").

If `record = TRUE`, returns a recorded plot object (single-group) or a named list of recorded plots (multi-group) created with `grDevices::recordPlot()`. You can later replay them with `grDevices::replayPlot()`.

## Value

- If `record = FALSE` (default): invisibly returns `NULL` (plots are drawn as a side-effect).
- If `record = TRUE`: a `recordedplot` (single-group) or a named `list` of `recordedplots` (multi-group).

## See Also

`resid_cor`,' `resid_qq`, `hopper_plot`, `corrplot.mixed`,

## Examples

```
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(HS.model, data = lavaan::HolzingerSwineford1939)

# Draw Bentler-type residual correlations
resid_corrplot(fit, type = "cor.bentler", order = "hclust")

# Draw standard residual correlations
resid_corrplot(fit, type = "cor")

# Capture plot object for later replay
rec <- resid_corrplot(fit, type = "cor.bentler", order = "hclust", record = TRUE)
if (interactive()) grDevices::replayPlot(rec)

# Multi-group demo of common_scale and title_prefix
fit_mg <- lavaan::cfa(HS.model, data = lavaan::HolzingerSwineford1939, group = "school")
# harmonized color scale across groups
resid_corrplot(fit_mg, type = "cor.bentler", common_scale = TRUE,  title_prefix = "School: ")
# per-group color scales
resid_corrplot(fit_mg, type = "cor.bentler", common_scale = FALSE, title_prefix = "School: ")
```

---

`resid_qq` *Q-Q plot of residual correlation z-statistics*

---

## Description

Draws a Q-Q plot for residual correlation z-statistics returned by `resid_cor()`. For multi-group models, a separate panel is drawn for each group. The `n` most extreme pairs (by |z|) are labeled.

## Usage

```
resid_qq(fit, n = 5, title = NULL)
```

## Arguments

| | |
|---|---|
| fit | A fitted lavaan object. |
| n | Number of most extreme \|z\| points to label (per group). Default 5. |
| title | Optional plot title. |

## Details

The z-statistics are expected to follow approximately a standard normal distribution N(0, 1) under correct model specification, so systematic deflections from the diagonal in the Q-Q plot indicate potential model misfit or localized residual dependencies.

If z is not available from lavaan::lavResiduals(), the function attempts to compute it as cor / se. If neither z nor se are available, the function stops with an informative error.

## Value

A ggplot2 object.

## See Also

resid_cor, resid_corrplot, hopper_plot

## Examples

```
# Single-group example
HS.model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
fit <- lavaan::cfa(HS.model, data = lavaan::HolzingerSwineford1939)
resid_qq(fit, n = 5)

# Multi-group example (groups by "school")
fit_mg <- lavaan::cfa(HS.model,
                      data  = lavaan::HolzingerSwineford1939,
                      group = "school")
resid_qq(fit_mg, n = 7, title = "Residual z Q-Q by group")
```

# Index