# Package: ackwards (via r-universe)

July 24, 2026

**Title** Bass-Ackwards Hierarchical Structural Analysis

**Version** 0.1.1

**Description** Implements Goldberg's (2006) <doi:10.1016/j.jrp.2006.01.001> bass-ackwards method and modern descendants for hierarchical structural analysis. Extracts solutions from 1 to k factors using principal component analysis (PCA), exploratory factor analysis (EFA), or exploratory structural equation modeling (ESEM) engines, then characterizes the hierarchy via between-level factor-score correlations computed via exact linear algebra (Waller, 2007, <doi:10.1016/j.jrp.2006.08.005>) or materialized scores. Includes the Forbes (2023) <doi:10.1037/met0000546> extension for redundancy pruning and all-levels cross-correlations.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** cli, generics, psych, rlang, stats, utils

**Suggests** covr, EFAtools, future, future.apply, ggplot2, gt, knitr, lavaan (>= 0.6-13), rmarkdown, testthat (>= 3.0.0)

**LazyData** true

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Config/testthat/start-first** suggest_k, boot_edges, esem, layout, missing, vignette-m24

**VignetteBuilder** knitr

**URL** https://jmgirard.github.io/ackwards/, https://github.com/jmgirard/ackwards

**BugReports** https://github.com/jmgirard/ackwards/issues

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Jeffrey M. Girard [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7359-3746>), Miriam K. Forbes
[cph] (Author of the bundled AMH correlation matrix
(data/forbes2023.rda), CC-BY 4.0)

**Maintainer** Jeffrey M. Girard <me@jmgirard.com>

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

**Date/Publication** 2026-07-24 10:00:13 UTC

**RemoteUrl** https://github.com/cran/ackwards

**RemoteRef** HEAD

**RemoteSha** 65b3be1a257580c9e51ea86712882fb0e117f436

# Contents

ackwards *Bass-ackwards hierarchical structural analysis*

## Description

Extracts factor/component solutions at levels 1 through k, then characterises the hierarchy by computing between-level factor-score correlations. The "hierarchy" is descriptive: edges are score correlations, not a fitted higher-order SEM.

## Usage

```
ackwards(
  data,
  k_max,
  engine = "pca",
  cor = "pearson",
  fm = "minres",
  estimator = NULL,
  missing = "pairwise",
  n_obs = NULL,
  align_signs = TRUE,
  keep_scores = FALSE,
  keep_fits = FALSE,
  seed = NULL,
  pairs = "adjacent",
  cut_show = 0.3,
  correct = 0.5,
  ...
)
```

## Arguments

| | |
|---|---|
| data | A data frame or numeric matrix of observed variables (items in columns, observations in rows). Alternatively, a pre-computed **correlation matrix** may be supplied (a square, symmetric, numeric matrix with unit diagonal). When a correlation matrix is supplied, engine must be "pca" or "efa" (ESEM requires raw data), and the missing and cor arguments are ignored. See the *Correlation-matrix input* section below. |
| k_max | Maximum number of factors/components to extract. Required; use suggest_k() if uncertain. Sets the *depth* of the hierarchy: levels 1 through k_max are all extracted and retained. (Note: suggest_k()'s k_max means something related but distinct – the max number of factors/components it *evaluates* when recommending a depth, not a depth itself; see that function's docs.) |
| engine | Extraction engine: "pca" (default), "efa", or "esem". "esem" uses lavaan::efa() with rotation-aware SEs and per-level fit indices; recommended for the clinical/HiTOP workflow (Kim & Eaton, 2015; Forbush et al., 2024). Requires lavaan >= 0.6-13. |

`cor` Correlation basis: `"pearson"` (default), `"spearman"`, or `"polychoric"`. For PCA/EFA, `"polychoric"` computes a polychoric correlation matrix via `psych::polychoric()` (requires psych). For ESEM, it triggers WLSMV estimation via lavaan. A cli warning is emitted when ordinal-looking columns are detected and `cor` is not `"polychoric"`.

`fm` Factor extraction method passed to `psych::fa()`; only used when `engine = "efa"`. One of `"minres"` (default, robust OLS), `"ml"` (maximum likelihood, gives chi-square fit but converges less reliably at deep levels), or `"pa"` (principal axis). Ignored for `engine = "pca"`.

`estimator` Estimation method for the ESEM engine. `NULL` (default) auto-selects: `"WLSMV"` when `cor = "polychoric"`, `"ML"` otherwise. Pass explicitly to override: `"ULSMV"` (unweighted WLS), `"MLR"` (robust ML). `cor = "polychoric"` with `estimator = "ML"/"MLR"` errors (lavaan itself does not support ML/MLR on ordered indicators); `"WLSMV"/ "ULSMV"` with a continuous `cor` is allowed (a valid, if atypical, continuous WLS/ADF estimator). Ignored for PCA and EFA engines. The effective value (after auto-selection) is recorded in `x$meta$estimator` (`NA` for PCA/EFA).

`missing` How to handle missing item responses. One of:

- `"pairwise"` (default) – use all available observations pairwise. For PCA/EFA this feeds `stats::cor(use = "pairwise.complete.obs")`. For ESEM with WLSMV/ULSMV (ordinal), lavaan uses `available.cases`, which computes polychoric thresholds and correlations from all rows that contribute to each pair – MCAR-valid and uses the full N. For ESEM with ML/MLR (continuous), lavaan uses listwise deletion internally while edges are computed from a pairwise correlation matrix; this minor inconsistency is documented in `$meta`. A warning is emitted when incomplete rows are detected.
- `"listwise"` – only complete rows are used. Reduces data to `stats::complete.cases()` before fitting, so the correlation matrix, the engine fit, and the edges are all consistent. `n_obs` in the result reflects the reduced N.
- `"fiml"` – Full Information Maximum Likelihood. For `engine = "esem"` (with `estimator = "ML"/"MLR"`), passes `missing = "fiml"` to `lavaan::efa()` and derives edge correlations from lavaan's FIML-estimated saturated model. For `engine = "pca"/"efa"` (M38), the correlation matrix is estimated via `psych::corFiml()` and fed to the usual `W'RW` algebra; this requires `cor = "pearson"` (corFiml estimates a multivariate-normal matrix) and the route is announced via a cli message. Errors for WLSMV/ULSMV and for a non-Pearson PCA/EFA basis. Note: FIML improves estimation under missingness but does not impute item responses; score materialisation (`keep_scores = TRUE`) still produces `NA` rows for incomplete observations. See `n_obs` for the fit-index sample size on the PCA/EFA path.

`n_obs` Number of observations, or (on the raw-data FIML path) a string selecting which N feeds the fit indices.

- **Correlation-matrix input:** a positive integer. Required for `engine = "efa"` (`psych::fa()` needs N for chi-square / RMSEA / TLI); optional for `"pca"` (stored as `NA_integer_` if omitted). For PCA it is recorded in the result

metadata and feeds the N-based sampling-adequacy checks only – PCA level fit is eigenvalue-based and computes no N-dependent fit statistics.
- **Raw data:** N is normally taken from `nrow(data)` and a numeric `n_obs` is ignored (with a warning). The exception is `missing = "fiml"` with `engine = "pca"/"efa"` (M38): because `psych::corFiml()` estimates the correlation matrix from incomplete rows, `n_obs` may be `"total"` (default – every row contributing to the FIML likelihood, matching the FIML convention; Enders, 2010) or `"complete"` (complete-case N, a conservative lower bound). Point estimates (loadings, edges) do not depend on this choice; only the EFA fit indices do, and those are *approximate* under this two-step (FIML matrix into normal-theory EFA) route regardless of N (Zhang & Savalei, 2020). A string `n_obs` is accepted only on this path.

`align_signs` Logical; sign-align factors to primary-parent lineage? Default `TRUE`.

`keep_scores` Logical; store factor scores in the result? Default `FALSE` (recomputable via `augment.ackwards()`). When `TRUE`, per-observation scores are stored in `x$scores` as a named list of `n x k_j` matrices, one per level, standardized by real score SDs (see `augment.ackwards()`).

`keep_fits` Logical; store raw engine fit objects? Default `FALSE`. When `TRUE`, the per-level fit objects (psych or lavaan) are stored in `x$fits` as a named list indexed by level.

`seed` Integer seed for stochastic engines (not used by PCA but captured for reproducibility metadata). Default `NULL`.

`pairs` Which level pairs to compute edges for: `"adjacent"` (default, classic Goldberg – only consecutive levels) or `"all"` (Forbes extension – every pair of levels, including skip-level correlations). `prune()` recomputes its own all-pairs edges on demand regardless of this setting, so pruning does not require `pairs = "all"` here.

`cut_show` Edges with `|r| >= cut_show` are flagged `above_cut` in `tidy()` output. Default `0.3`.

`correct` Continuity correction passed to `psych::polychoric()` on the PCA/EFA polychoric path (`engine = "pca"/"efa"` with `cor = "polychoric"`). Default `0.5` (psych's own default), which adds that value to zero cells before estimating thresholds. **Set** `correct = 0` if `psych::polychoric()` fails on your data (its error suggests exactly this) – this typically happens when an item has a near-empty response category or when items with unequal category counts produce a sparse cross-cell. Ignored on other paths: ESEM computes its own polychoric correlations inside lavaan, and the Pearson/Spearman bases do not use it.

`...` Reserved for future arguments.

## Value

An object of class `"ackwards"`. See `print.ackwards()`, `tidy.ackwards()`, `glance.ackwards()`, and `augment.ackwards()` for output methods.

## Defaults and why

- `engine = "pca"` – the original Goldberg (2006) method; fastest; never fails to converge; the Waller (2007) algebra is exact for components.

- `rotation = "varimax"` – the `T'=T^-1` property of orthogonal rotation enables the closed-form `W'RW` edge algebra and keeps within-level factors uncorrelated so cross-level edges reflect only the hierarchical signal. Matches Goldberg (2006), Kim & Eaton (2015), and Forbush et al. (2024). Varimax is the only supported rotation; oblique rotation would confound the between-level signal that is the method's core output.
- `cor = "pearson"` – no silent basis switching. If your items look ordinal (<= 7 distinct integer values), a cli warning will suggest `cor = "polychoric"`, which is available for all three engines.
- `align_signs = TRUE` – unaligned signs make the output unreadable. Anchor: m1f1 is oriented toward the positive manifold; each subsequent factor is flipped so its edge to its primary parent is positive.
- `keep_scores = FALSE` / `keep_fits = FALSE` – memory and privacy. Scores are O(n x Sigmak) and often sensitive; raw engine fits can be large. Both are recomputable from the stored `r` matrix.

### Performance (ESEM, large item sets)

The ESEM engine fits a separate `lavaan` model at every level 1..`k_max`. For ordinal data (`cor = "polychoric"`, WLSMV) the costly sample statistics lavaan derives from the raw data – thresholds, the polychoric correlation matrix, and the asymptotic weight matrix – depend only on the data, not on the number of factors, so they are **computed once** at the first level and **reused** for every deeper level (identical solutions, much less work). This matters most when you have many items (hundreds), where recomputing those statistics at each level dominated the run time.

The per-level model fits are mutually independent and are dispatched through the **future** framework when **future.apply** is installed. By default the plan is sequential (no behaviour change). To run the levels in parallel, set a plan once before calling `ackwards()`:

```
future::plan(future::multisession, workers = 4)  # or multicore on Unix
x <- ackwards(items, k_max = 8, engine = "esem", cor = "polychoric")
```

Parallelism pays off when the per-level fits are heavy (large `p`, several levels); for small problems the worker startup cost can outweigh it. Results are reproducible across plans when `seed` is supplied. PCA and EFA already compute their correlation matrix once and are unaffected.

### Correlation-matrix input

When `data` is a pre-computed correlation matrix (square, symmetric, unit diagonal), `ackwards()` runs entirely from that matrix using the `W'RW` algebra – no raw item responses are needed. This is useful when you have a published correlation table or a polychoric matrix computed externally.

Constraints and behaviour when a correlation matrix is supplied:

- **Engine:** only `"pca"` and `"efa"` are supported. `"esem"` requires raw data (for lavaan's own polychoric computation, WLSMV estimation, and per-level fit indices) and will error clearly.
- `n_obs`**:** required for `"efa"` (psych needs N for chi-square / RMSEA / TLI); optional for `"pca"` (stored as `NA` if omitted; used for the N-based sampling-adequacy checks and result metadata only – PCA computes no N-dependent fit statistics).
- `cor` **argument:** ignored – the basis is already determined by the matrix you supply. A warning is emitted if you set `cor` explicitly.

- `missing` **argument:** ignored – missingness was handled when computing the matrix. A warning is emitted if you set `missing` explicitly.
- **Factor scores:** `keep_scores = TRUE` will error; `augment()` and `tidy(what = "scores")` will also error because individual-level scores require row-level item responses.
- `$cor` **field:** stored as `NA_character_`; printed as `"(user-supplied matrix)"`.

## When to trust the result

`ackwards()` raises diagnostics as it fits. They fall into three tiers by what they mean for whether you should trust and report the solution:

**Fatal – fix before trusting.** The result is undefined or rests on a broken correlation matrix:

- A **constant item** (no variance) errors – drop it (see `check_items()`).
- `psych::polychoric()` **fails** – usually a near-empty response category; set `correct = 0` or collapse rare categories.
- A **level fails to converge** – the hierarchy is truncated to the deepest level that did; do not interpret beyond it.
- A **near-singular correlation matrix** (smallest eigenvalue < `1e-4`, recorded in `meta$near_singular` / `meta$min_eigenvalue` and re-surfaced by `print()`/`summary()`) means per-level fit indices and factor scores are unreliable and the loadings/edges rest on a rank-deficient matrix. (For EFA the residual-based fallback inflates `TLI`/`RMSEA`; for ESEM `CFI` comes back `NA`.) Trim redundant items, use `missing = "listwise"`, or – on the polychoric basis – collapse sparse categories or set `correct = 0`.

**Caution – interpret carefully.** The solution exists but may be unstable:

- A **Heywood case** (communality `> 1` / negative uniqueness) at a level – check that level's loadings make substantive sense; consider fewer factors.
- A **near-constant item** (one response category dominates) can drive a meaningless factor – inspect it with `check_items()`.
- **Ordinal data on a Pearson basis** attenuates correlations – fine for a quick look or `suggest_k()` screening, but report the final model on `cor = "polychoric"`.

**Informational – usually fine.** Proceed, just be aware: the pairwise-missing note, a merely *sparse* (rare-but-present) response category, and the ordinal-detection warning when you did intend Pearson.

## References

Goldberg, L. R. (2006). Doing it all Bass-Ackwards: The development of hierarchical factor structures from the top down. *Journal of Research in Personality*, 40(4), 347–358. doi:10.1016/j.jrp.2006.01.001

Waller, N. G. (2007). A general method for computing hierarchical component structures by Goldberg's bass-ackwards method. *Journal of Research in Personality*, 41(4), 745–752. doi:10.1016/j.jrp.2006.08.005

Forbes, M. K. (2023). Improving hierarchical models of individual differences: An extension of Goldberg's bass-ackward method. *Psychological Methods*. doi:10.1037/met0000546

Enders, C. K. (2010). *Applied Missing Data Analysis*. Guilford Press.

Zhang, X., & Savalei, V. (2020). Examining the effect of missing data on RMSEA and CFI under normal theory full-information maximum likelihood. *Structural Equation Modeling*, 27(2), 219–239. doi:10.1080/10705511.2019.1642111

## See Also

`print.ackwards()`, `tidy.ackwards()`, `glance.ackwards()`, `prune()` (Forbes-extension redundancy/artifact flagging, piped off the result of this function)

## Examples

```
# sim16 is continuous with a known 1 -> 2 -> 4 hierarchy, so the default
# pearson basis is appropriate. For ordinal items (e.g. bfi25), fit on the
# polychoric basis instead -- see `cor = "polychoric"` and the ordinal
# vignette.
x <- ackwards(sim16, k_max = 4)
print(x)
tidy(x)
glance(x)

# Correlation-matrix input (PCA engine; n_obs optional)
R <- cor(sim16)
x_R <- ackwards(R, k_max = 4)
print(x_R)
```

---

# augment.ackwards *Augment data with factor scores from an ackwards object*

---

## Description

Appends per-observation factor scores for every level to a data frame. Score columns are named `.m{k}f{j}` (e.g., `.m1f1`, `.m2f1`, `.m2f2`), matching the factor labels used throughout the object.

## Usage

```
## S3 method for class 'ackwards'
augment(
  x,
  data = NULL,
  append = TRUE,
  id_cols = NULL,
  scaling = c("fit", "sample"),
  ...
)
```

## Arguments

| | |
|---|---|
| x | An ackwards object. |
| data | A data frame or numeric matrix with the same variables (columns) used to fit x. When NULL (default), uses pre-stored scores if available (requires keep_scores = TRUE at fit time). |
| append | Logical. When TRUE (default) the score columns are appended to data (or to a .obs index when data is NULL). When FALSE only the score columns are returned (plus any id_cols). |
| id_cols | Optional character vector naming columns of data to carry through alongside the scores when append = FALSE (for example a subject identifier, so scores can be rejoined after filtering). Ignored – and an error – when append = TRUE (all columns are already kept) or when data is NULL (there are no source columns to carry). NULL (default) returns the bare score columns. |
| scaling | Which item means/SDs standardize data before the weights are applied. "fit" (default) uses the **fit-time** moments stored in the object – the correct choice for scoring new observations (e.g. a cross-validation test split) or subsets on the training metric. "sample" standardizes by the supplied data's own moments (the only option for objects fit from a correlation matrix, which carry no raw-data moments). Only used when data is supplied; passing it without data is an error (stored scores are returned exactly as computed at fit time). |
| ... | Ignored. |

## Details

**Score computation.** Scores are `S = Z W / sqrt(score_var)`, where `Z` is the item z-scores, `W` is the per-level weight matrix stored in the object, and `sqrt(score_var)` standardizes by the real score standard deviations (Invariant 1: never assume unit variance). For PCA the method is `"components"`; for EFA/ESEM it is `"tenBerge"`. The `scaling` argument controls which means/SDs build `Z`: by default the **fit-time** moments stored in the object, so any data you score — the training data, a subset of it, or entirely new observations — lands on the same metric the model was estimated in.

**Scoring new observations (cross-validation).** Because scoring only needs the stored weight matrices and the fit-time moments, you can fit `ackwards()` on a training split and score a held-out test split *without retraining*: `augment(x, data = test_set)` (or, equivalently, `predict.ackwards()`). Under the default `scaling = "fit"` the test observations are standardized by the *training* means/SDs, which is what "applying the trained model" means: a test observation's score does not depend on which other observations happen to share its split, and train and test scores are directly comparable. `scaling = "sample"` instead re-standardizes by the supplied data's own moments — a deliberate choice when scoring a sample from a different population in its own metric, and the only option for objects fit from a correlation matrix (which carry no raw-data moments). For non-Pearson bases (polychoric, Spearman) the usual caveat applies either way: the weights derive from the non-Pearson `R` while `Z` is a linear standardization, so empirical score SDs are close to but not exactly 1 (a one-time warning says so); train/test comparability under `scaling = "fit"` is unaffected. For objects fit with `missing = "fiml"` (PCA/EFA), the stored moments are the observed (`na.rm`) means/SDs of the training items — the correlation matrix was FIML-estimated, but scoring operates on observed responses, so the observed moments are the consistent frame; incomplete rows still score `NA` (scoring does not impute).

**Missing data.** Score projection applies weights row-wise and propagates NAs listwise: any observation with at least one missing item variable will produce NA scores at every level. This differs from fitting, which uses pairwise-complete correlations. A warning is issued if NA rows are detected. Use `na.omit(data)` before scoring if NA rows are unwanted.

**Data source.** If `data` is supplied, scores are always recomputed from it using the stored weights – this is how to score new observations. If `data` is `NULL` and `keep_scores = TRUE` was set at fit time, the stored scores are returned. If neither is available an informative error is raised.

**Scores-only output (`append = FALSE`).** By default (`append = TRUE`) the scores are appended to the supplied `data`, following the broom convention. Set `append = FALSE` to return *only* the score columns – convenient for feeding scores straight into `cor()`, `lm()`, or a clustering call without dragging the item columns along. Because `augment()` always preserves row order and row count, `cbind(data, augment(x, data, append = FALSE))` reproduces the appended output exactly. A row that scores NA (because it is missing an item – see *Missing data* above) is still returned in place, so the positional alignment holds even with NA scores. For a rejoin that survives *filtering* the scores afterwards, name identifier columns with `id_cols` so they travel with the scores.

## Value

A data frame. With `append = TRUE`: the supplied `data` (or a `.obs` index when `data` is `NULL`) with score columns appended. With `append = FALSE`: only the score columns, optionally prefixed by the `id_cols`. Row order and count always match the input.

## See Also

`ackwards()`, `tidy.ackwards()`

## Examples

```
# Score the training data on the fly (no keep_scores = TRUE needed)
x <- ackwards(sim16, k_max = 5)
scores_df <- augment(x, data = sim16)
head(scores_df[, startsWith(names(scores_df), ".m")])

# Scores-only: just the .m{k}f{j} columns, ready for cor()/lm()
scores_only <- augment(x, data = sim16, append = FALSE)
round(cor(scores_only[, c(".m5f1", ".m5f2")]), 2)

# Carry an identifier through for a safe post-filter rejoin
df <- data.frame(id = seq_len(nrow(sim16)), sim16)
scored <- augment(x, data = df, append = FALSE, id_cols = "id")
head(scored[, c("id", ".m5f1")])

# Store at fit time and augment without re-supplying data
x2 <- ackwards(sim16, k_max = 5, keep_scores = TRUE)
scores_df2 <- augment(x2)

# Cross-validation: fit on a training split, score the test split on the
# training metric (no retraining; see also predict.ackwards())
train_idx <- seq_len(500)
x_train <- ackwards(sim16[train_idx, ], k_max = 5)
```

```
test_scores <- augment(x_train, data = sim16[-train_idx, ], append = FALSE)
head(test_scores)
```

---

`autoplot` *autoplot generic*

---

## Description

Create ggplot2-based visualisations from model objects.

This generic is defined in **ackwards** so that `autoplot.ackwards()` is available without requiring `library(ggplot2)`. When **ggplot2** is also loaded its own `autoplot` generic takes over in the search path, but S3 dispatch still finds `autoplot.ackwards` correctly via either route. Defining our own generic is intentional – it avoids putting ggplot2 in Imports while keeping the `autoplot(x)` ergonomic without a `library()` call.

## Usage

```
autoplot(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object to visualise. |
| `...` | Additional arguments passed to methods. |

## Value

A `ggplot` object.

---

`autoplot.ackwards` *Plot a bass-ackwards diagram or per-level fit index chart*

---

## Description

When `what = "hierarchy"` (default), renders the layered bass-ackwards hierarchy as a ggplot2 diagram. Factors appear as labelled boxes arranged in levels (level 1 at top, level k at bottom by default; set `direction = "horizontal"` for a left-to-right layout). Between-level edges below `cut_show` are hidden. Two aesthetics carry the edge information, each explained by its own legend: **sign** (positive/negative) is shown by `sign_by` (edge colour by default) and **magnitude** (`|r|`) by `magnitude_by` (line thickness by default). No aesthetic is ever mapped without a matching legend.

## Usage

```
## S3 method for class 'ackwards'
autoplot(
  object,
  what = c("hierarchy", "fit"),
  sign_by = c("color", "linetype", "both", "none"),
  magnitude_by = c("linewidth", "none"),
  cut_show = NULL,
  color_pos = "#2166AC",
  color_neg = "#D6604D",
  color_edge = "black",
  colour_pos = NULL,
  colour_neg = NULL,
  colour_edge = NULL,
  node_width = 0.8,
  node_height = 0.4,
  min_sep = 1,
  show_skip = NULL,
  curvature = 0.2,
  color_pruned = "grey80",
  colour_pruned = NULL,
  show_r = FALSE,
  r_digits = 2L,
  r_label_size = 2.5,
  mono = FALSE,
  edge_linewidth = NULL,
  cut_strong = NULL,
  direction = c("vertical", "horizontal"),
  show_level_labels = TRUE,
  level_label_size = 3,
  node_labels = NULL,
  primary_only = FALSE,
  drop_pruned = FALSE,
  compress_levels = FALSE,
  show_arrows = TRUE,
  legend = TRUE,
  ...
)

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

## Arguments

| | |
|---|---|
| object | An ackwards object. |
| what | One of "hierarchy" (default) or "fit". Controls which visualisation is produced. All other parameters are ignored when what = "fit". |

`sign_by` How edge **sign** (positive vs negative correlation) is encoded. One of `"color"` (default; positive = `color_pos`, negative = `color_neg`), `"linetype"` (positive = solid, negative = dashed), `"both"` (colour *and* linetype, with negative drawn as a distinct double-dash so it reads clearly in greyscale), or `"none"` (sign not encoded; all edges `color_edge` and solid). Whichever channel is used gets a "Direction" legend. Colour is the default because it reads sign pre-attentively and leaves linetype free; switch to `"linetype"` or `"both"` for greyscale or colour-blind-safe figures.

`magnitude_by` How edge **magnitude** (`|r|`) is encoded. One of `"linewidth"` (default; thicker = stronger, with a `|r|` legend) or `"none"` (constant width). A numeric `edge_linewidth` also forces constant width at that value.

`cut_show` Edges with `|r| < cut_show` are not drawn. Defaults to the value used when the object was created (`x$meta$cut_show`, typically 0.3).

`color_pos`, `colour_pos`
Colour for positive edges when `sign_by` uses colour. Default `"#2166AC"` (blue). `colour_pos` is an accepted British alias.

`color_neg`, `colour_neg`
Colour for negative edges when `sign_by` uses colour. Default `"#D6604D"` (red). `colour_neg` is an accepted British alias.

`color_edge`, `colour_edge`
Single colour for all edges when `sign_by` does not use colour (`"linetype"` or `"none"`). Default `"black"`. `colour_edge` is an accepted British alias.

`node_width` Width of factor boxes in layout units. Default `0.8`.

`node_height` Height of factor boxes in layout units. Default `0.4`.

`min_sep` Minimum horizontal separation between nodes; passed to `ba_layout()`. Default `1.0`.

`show_skip` Whether to draw skip-level (non-adjacent) edges. `NULL` (default) auto-detects: `TRUE` when the object was run with `pairs = "all"`, `FALSE` otherwise. Ignored when `drop_pruned = TRUE`.

`curvature` Curvature of skip-level edge arcs. Passed to `ggplot2::geom_curve()`. Positive values curve right; default `0.2`. Ignored when `drop_pruned = TRUE`.

`color_pruned`, `colour_pruned`
Fill colour for nodes flagged as pruned/redundant. Default `"grey80"`. Only applied when the object carries pruning annotations (`x$prune` is non-`NULL`) and `drop_pruned = FALSE` (pruned nodes are omitted entirely when `drop_pruned = TRUE`). `colour_pruned` is an accepted British alias.

`show_r` Whether to label each drawn edge with its correlation coefficient. Default `FALSE`. When `TRUE`, labels are formatted in APA style (leading zero stripped, e.g. `.23`, `-.30`) and placed beside each edge using a white-background label that clears the arrowhead.

`r_digits` Number of decimal places for edge labels when `show_r = TRUE`. Default `2L`.

`r_label_size` Font size for edge correlation labels when `show_r = TRUE`. Passed to `ggplot2::geom_label()` as `size`. Default `2.5`.

`mono` Monochrome convenience wrapper. When `TRUE`, equivalent to `sign_by = "linetype"` with black edges (it overrides `sign_by` and the colour arguments). `magnitude_by` still applies. Default `FALSE`.

`edge_linewidth` `NULL` (default) uses `magnitude_by` to decide edge width. A numeric value forces every edge to that constant width (implying `magnitude_by = "none"`) and drops the `|r|` legend. Forbes figures use uniform thin lines (~= 0.5–0.6).

`cut_strong` **Deprecated** in M35 and ignored (with a warning). Edge magnitude is now shown by `magnitude_by` (linewidth) and sign by `sign_by`; the old strong/weak linetype split double-encoded magnitude. Retained only so existing calls do not error.

`direction` Layout orientation. `"vertical"` (default) stacks levels top-to-bottom (level 1 at top); `"horizontal"` lays them out left-to-right (level 1 at left), which suits wide slides or posters. Level-axis labels move to the bottom margin under `"horizontal"`.

`show_level_labels` Whether to draw level axis labels ("1 factor", "2 factors", ...) to the left of the diagram. Default `TRUE`. When the object carries pruning annotations (`x$prune` non-`NULL`) and `drop_pruned = FALSE`, a level whose factors are *all* pruned has its axis label rendered in italic to denote its status (matching the grey node fill); partially-pruned levels keep a plain label.

`level_label_size` Font size for level axis labels. Default `3`.

`node_labels` A named character vector mapping factor IDs (e.g. `"m5f1"`) to custom display strings (e.g. `"General"`). Unspecified factors keep any label attached with `set_factor_labels()`, falling back to the default `m{k}f{j}` ID. Entries here **override** a stored factor label for that node, so this argument is a per-call last word over the persistent labels. A warning is issued for names that match no factor ID in the object. Default `NULL`.

`primary_only` When `TRUE`, only primary-parent edges (`is_primary == TRUE`) are drawn. Because skip-level edges are never primary, this also suppresses skip arcs. Ignored when `drop_pruned = TRUE`. Default `FALSE`.

`drop_pruned` When `TRUE`, activates the Forbes (2023) pruned-view rendering path: pruned nodes are removed from the diagram entirely and each retained node is connected to its single strongest kept ancestor by a straight arrow (even across level gaps). Requires the object to carry pruning annotations (piped through `prune()`); errors if not. Overrides `show_skip`, `curvature`, and `primary_only`. Default `FALSE`.

`compress_levels` When `TRUE` under `drop_pruned = TRUE`, closes vertical gaps left by pruned levels so retained levels are evenly spaced; level axis labels (d) still show the original level numbers. Ignored when `drop_pruned = FALSE`. Default `FALSE`.

`show_arrows` When `FALSE`, edges are drawn with plain line ends instead of closed arrowheads (`arrow = NULL`). Applies to both straight and curved edge layers. Default `TRUE`. Forbes (2023) figures use plain line ends.

`legend` When `FALSE`, suppresses all plot legends (`legend.position = "none"`). Useful when `color_pos == color_neg` (e.g. both `"black"`) to remove an otherwise redundant Direction key. Default `TRUE`.

| | |
|---|---|
| ... | Ignored. |
| x | An ackwards object. |

## Details

When what = "fit", renders a two-panel line plot of per-level fit indices (CFI/TLI in the top panel; RMSEA/SRMR in the bottom panel) with horizontal reference lines at conventional Hu & Bentler (1999) thresholds. The anchor level (k = 1, saturated and always fits perfectly) is excluded. Requires an EFA or ESEM engine; returns an informative empty plot for PCA (which has no model-fit indices).

Requires the **ggplot2** package.

## Value

A ggplot object.

## Saving plots

autoplot() returns a standard ggplot object, so save it with ggplot2::ggsave():

```
p <- autoplot(x)
ggplot2::ggsave("hierarchy.png", p, width = 8, height = 6, dpi = 300)
```

ggsave() is not re-exported by **ackwards** (that would move **ggplot2** from Suggests into Imports); call it from **ggplot2** directly. For a wide slide or poster, pair it with direction = "horizontal".

## See Also

ba_layout(), plot.ackwards()

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # sim16 has a known 1 -> 2 -> 4 hierarchy (continuous; pearson basis)
  x <- ackwards(sim16, k_max = 5)
  autoplot(x)
  autoplot(x, color_pos = "steelblue")

  # Encode sign by linetype instead of colour, or by both
  autoplot(x, sign_by = "linetype")
  autoplot(x, sign_by = "both")

  # Left-to-right layout (wide slides / posters)
  autoplot(x, direction = "horizontal")

  # Per-level fit index chart (EFA or ESEM only)
  x_efa <- ackwards(sim16, k_max = 5, engine = "efa")
  autoplot(x_efa, what = "fit")

  # Monochrome with correlation labels (for greyscale figures)
  autoplot(x, mono = TRUE, show_r = TRUE)
```

```
  # Custom node labels for the 5-factor level
  autoplot(x, node_labels = c(m5f1 = "Factor A", m5f2 = "Factor B"))

  # Primary links only -- clean hierarchy tree
  autoplot(x, primary_only = TRUE)

  # Forbes pruned view: omit redundant nodes, straight spanning arrows
  xp <- ackwards(sim16, k_max = 5) |> prune("redundant")
  autoplot(xp, drop_pruned = TRUE)
  autoplot(xp, drop_pruned = TRUE, show_r = TRUE)
  autoplot(xp, drop_pruned = TRUE, compress_levels = TRUE)

  # Plain line ends without arrowheads
  autoplot(x, show_arrows = FALSE)

  # Uniform edge width (no |r| scaling)
  autoplot(x, edge_linewidth = 0.5)

  # Suppress legend
  autoplot(x, legend = FALSE)

  # Forbes (2023) publication style: black lines, uniform width, no arrowheads
  autoplot(xp,
    drop_pruned = TRUE,
    color_pos = "black", color_neg = "black",
    edge_linewidth = 0.6, show_arrows = FALSE, legend = FALSE
  )
}
```

---

`autoplot.comparability` *Plot a comparability diagnostic*

---

## Description

Renders a two-panel ggplot2 diagnostic for a `comparability()` object: score comparability (r) and loading congruence (Tucker's phi) for every factor at every level. Grey points are individual splits; black points are the per-factor medians. Dashed and dotted reference lines mark the conventional .90 / .95 benchmarks – visual guides, not tests.

## Usage

```
## S3 method for class 'comparability'
autoplot(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | A `comparability` object. |
| `...` | Ignored. |

### Details

Requires the **ggplot2** package.

### Value

A `ggplot` object.

### See Also

`comparability()`

### Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  cmp <- comparability(sim16, k_max = 5, n_splits = 5, seed = 1)
  autoplot(cmp)
}
```

---

## `autoplot.suggest_k` *Plot a suggest_k diagnostic*

---

### Description

Renders a ggplot2 diagnostic for a `suggest_k` object. The plot shows one panel for each criterion group that was requested: Scree/PA (if `"pa_pc"` or `"pa_fa"` were requested), MAP (if `"map"`), VSS (if `"vss"`), and CD RMSE (if `"cd"` and **EFAtools** was available). When four panels are shown the layout is a 2x2 grid; otherwise a single-column layout is used. The recommended k for each criterion is marked with a star-shaped point.

### Usage

```
## S3 method for class 'suggest_k'
autoplot(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | A `suggest_k` object. |
| `...` | Ignored. |

## Details

The scree panel shows both PC and FA observed eigenvalues alongside their respective random-data thresholds (for whichever PA bases were requested). PA-PC compares the blue "Observed (PC)" line to the dashed PA-PC threshold; PA-FA compares the teal "Observed (FA)" line to the dotted PA-FA threshold.

The CD panel plots the mean RMSE between observed and comparison-data eigenvalues at each k. CD uses a sequential one-sided Wilcoxon test (Ruscio & Roche, 2012): a factor is retained while adding it significantly reduces RMSE (default $\alpha = 0.30$); the starred k is the last retained factor. The curve is shown only over the levels that were actually computed; the starred k need not be the visible minimum of the plotted curve.

Requires the **ggplot2** package.

## Value

A `ggplot` object.

## See Also

`suggest_k()`

## Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  sk <- suggest_k(sim16, n_iter = 5)
  autoplot(sk)
}
```

---

`ba_layout` *Compute a layered layout for a bass-ackwards diagram*

---

## Description

Returns tidy node coordinates and the edge table needed to render a bass-ackwards diagram. The layout uses a two-pass barycenter algorithm:

## Usage

```
ba_layout(x, min_sep = 1)
```

## Arguments

| | |
|---|---|
| `x` | An `ackwards` object. |
| `min_sep` | Minimum horizontal separation between adjacent nodes at the same level. Default `1.0`. Increase for wider diagrams. |

## Details

1. **Top-down pass** – determines the left-to-right *order* of factors at each level. Each factor's ordinal rank is the |r|-weighted mean of its parents' ranks in the level above, so siblings that share a parent are grouped together.
2. **Bottom-up pass** – assigns actual x coordinates. The deepest level (level k) is spread evenly; every upper-level factor is placed at the simple mean x of its **primary** children: a factor with one primary child lands directly above it; a factor with two primary children lands exactly halfway between them. Falls back to |r|-weighted mean of all children for factors with no primary children. Spreading resolves any remaining overlaps.

After both passes the layout is shifted so that the single level-1 node is always at x = 0.

## Value

A list with two data frames:

| | |
|---|---|
| `nodes` | One row per factor: `id`, `level`, `x`, `y`, `label`. `y = -level` so level 1 is at the top. |
| `edges` | The tidy edge table from `x$edges$tidy`. |

## See Also

`autoplot.ackwards()`, `tidy.ackwards()`

## Examples

```
x <- ackwards(sim16, k_max = 5)
lay <- ba_layout(x)
head(lay$nodes)
```

---

# bfi25 *Big Five Inventory – 25-item IPIP example dataset*

---

## Description

A 1 000-row subset of the 25 IPIP Big Five personality-marker items from Revelle's psych package (`psych::bfi`). Included so that package examples and vignettes run without reaching into psych's namespace directly.

## Usage

```
bfi25
```

## Format

A data frame with 1 000 rows and 25 integer columns (Likert responses scored 1–6, with some NAs reflecting genuine missing values in the original survey). The 25 items span the Big Five personality domains:

| Columns | Domain |
|---|---|
| A1–A5 | Agreeableness |
| C1–C5 | Conscientiousness |
| E1–E5 | Extraversion |
| N1–N5 | Neuroticism |
| O1–O5 | Openness |

## Details

Derived from `psych::bfi[, 1:25]` by sampling 1 000 rows with `set.seed(42)`. Missing values are preserved as in the original data. The full dataset (`psych::bfi`) contains responses from 2 800 participants collected via the SAPA project.

Each item column carries its public-domain IPIP stem (Goldberg, 1999) as a `label` attribute, so `ackwards()` captures it at fit time and `top_items()` prints the wording as `code: label` (e.g. `E4: Make friends easily`) with no setup. These are plain attributes: base row-subsetting (e.g. `na.omit(bfi25)`, `bfi25[rows, ]`) drops them, as base R does for any non-`labelled`-class vector, so fit on `bfi25` **directly** – its NAs are handled by the `missing` argument of `ackwards()` – to keep the labels.

To regenerate this dataset, run `source("data-raw/bfi25.R")` from the package root.

## Source

Revelle, W. (2026). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package. https://CRAN.R-project.org/package=psych

Data collected via the SAPA (Synthetic Aperture Personality Assessment) project (https://www.sapa-project.org/). Items are public-domain IPIP (International Personality Item Pool) markers developed by Goldberg (1999):

Goldberg, L. R. (1999). A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. In I. Mervielde, I. Deary, F. De Fruyt, & F. Ostendorf (Eds.), *Personality Psychology in Europe*, Vol. 7, pp. 7–28. Tilburg University Press.

## Examples

```
dim(bfi25)
head(bfi25)
attr(bfi25$E4, "label") # public-domain IPIP item stem
```

`boot_edges` *Bootstrap confidence intervals for between-level edges*

## Description

Attaches nonparametric bootstrap standard errors and percentile confidence intervals to every between-level correlation (edge) of a fitted bass-ackwards hierarchy. Every edge `ackwards()` reports is a point estimate; `boot_edges()` quantifies its sampling uncertainty, which matters most where a hard threshold consumes the estimate – `prune()`'s `|r| >= redundancy_r` redundancy rule, and the Forbes (2023) practice of interpreting the strongest all-pairs edge.

## Usage

```
boot_edges(x, ...)

## S3 method for class 'ackwards'
boot_edges(x, data, n_boot = 1000L, conf = 0.95, seed = NULL, ...)
```

## Arguments

| | |
|---|---|
| `x` | An `ackwards` object fit with `engine = "pca"` or `"efa"` on a `"pearson"` or `"spearman"` basis. ESEM objects are not supported (refitting `n_boot` lavaan hierarchies needs its own performance treatment), nor are polychoric-basis objects (estimating a polychoric matrix in every resample is slow and unstable) or objects fit from a correlation matrix (resampling needs rows). |
| `...` | Reserved for future arguments. |
| `data` | The raw item data the model was fit on. Required – the `ackwards` object deliberately does not store raw data (light core), so it must be re-supplied here. Columns are matched by name against the fit; a warning is issued if the data do not look like the fit data. |
| `n_boot` | Number of bootstrap replicates. Default `1000L`; larger values (2000+) are advisable for published interval endpoints (Efron & Tibshirani, 1993). Each replicate refits the full hierarchy, so cost scales linearly. |
| `conf` | Confidence level for the percentile intervals. Default `0.95`. |
| `seed` | Integer seed for reproducible resampling. `NULL` (default) uses the current RNG state. |

## Details

For each of `n_boot` replicates, `n` rows are resampled with replacement, the correlation matrix is recomputed with the same basis and missing-data routine used at fit time, and the full hierarchy is refit. Each replicate level is then **anchored to the full-sample solution** – its factors matched (greedy max-|r| with removal) and sign-oriented against the full-sample factors on the full-sample correlation matrix – before its edges are computed. Without anchoring, factor label switching and sign flipping across replicates would corrupt the pooled edge distributions; this is the same matching machinery `comparability()` uses.

All resample indices are drawn upfront from seed, so results are reproducible and identical whether replicates run serially or in parallel. Replicate fits are dispatched through **future.apply** when it is installed and the user has set a future::plan() (serial otherwise, as in ackwards()'s ESEM engine).

## Value

x, invisibly modified: the $boot element is populated with

edges — Data frame with one row per edge (aligned with tidy(x, what = "edges")): from, to, level_from, level_to, r (the full-sample point estimate), se (bootstrap standard error), lo, hi (percentile interval endpoints), and n_boot_ok (usable replicates for that edge).

n_boot, conf, seed — The request.

After calling boot_edges(), tidy(x, what = "edges") gains se, lo, and hi columns, and print(x)/summary(x) note the interval coverage.

## What the intervals do and do not fix

Per-edge intervals make sampling uncertainty **visible**: an edge whose interval straddles prune()'s redundancy_r threshold should not be treated as decisively above or below it. They do **not** correct the selection bias of scanning many edges for the strongest one – the maximum of hundreds of correlations capitalizes on chance even when every individual interval is honest. Treat the intervals as per-edge error bars, not a familywise inference.

## Failed replicates

A replicate whose hierarchy fails to converge (in full or at some levels) contributes NA to the affected edges and is dropped from their distributions – convergence is data, not an error. The usable replicate count is reported per edge in n_boot_ok, and a message summarises any shortfall.

## References

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall.

Forbes, M. K. (2023). Improving hierarchical models of individual differences: An extension of Goldberg's bass-ackward method. *Psychological Methods*. doi:10.1037/met0000546

## See Also

prune() for the thresholded rules the intervals contextualise, comparability() for split-half replicability of the factors themselves, tidy.ackwards() for the augmented edge table.

## Examples

```
x <- ackwards(sim16, k_max = 3)
x <- boot_edges(x, sim16, n_boot = 100, seed = 1)
x$boot$edges
head(tidy(x)) # now carries se / lo / hi
```

`check_items` *Screen items for problems before factor analysis*

## Description

Real-world item sets often contain columns that break or *silently* degrade a factor analysis: an item everyone answered the same way (no variance), an item dominated by one response with a couple of stray answers, or an item with heavy missingness. On the polychoric basis these are especially costly – a near-empty response category can make `psych::polychoric()` fail outright (see the `correct` argument of `ackwards()`), and a near-constant item can produce a plausible-looking but meaningless factor with no warning.

`check_items()` reports these problems *before* you fit, one row per item, so you can collapse rare categories, drop degenerate items, or set `cor`/`correct` deliberately rather than debugging a cryptic error. It only reports; it never changes your data. `ackwards()` runs the same screen internally: it **errors** on a constant item and **warns** on a near-degenerate one, naming the offenders.

## Usage

```
check_items(data, cor = c("polychoric", "pearson", "spearman"))
```

## Arguments

| | |
|---|---|
| `data` | A data frame or numeric matrix (items in columns). |
| `cor` | The correlation basis you plan to use: `"polychoric"` (default), `"pearson"`, or `"spearman"`. Only affects which problems are flagged – sparse response categories destabilise the polychoric basis but not the others. |

## Value

A data frame (class `check_items`) with one row per item and columns `item`, `n_valid`, `pct_missing`, `n_distinct`, `min_count` (smallest observed category count), `top_prop` (proportion of valid responses in the most common value), and `flag` – one of `"ok"`, `"constant"`, `"near-constant"`, `"sparse category"`, or `"high missing"`. Print it for a grouped summary with guidance; treat it as a plain data frame for the full per-item table.

## See Also

`ackwards()` and its `correct` argument (the polychoric failure this screens for); `factorability()`, `suggest_k()`, and `comparability()`, the other pre-analysis diagnostics.

## Examples

```
# sim16 is clean continuous data -> nothing flagged
check_items(sim16, cor = "pearson")

# An item dominated by one response with a couple of stray answers is flagged
d <- sim16
d$bad <- c(rep(1L, nrow(d) - 2L), 2L, 3L)
check_items(d)
```

---

`comparability` *Split-half factor comparability*

---

## Description

Measures how well each factor at each level of a bass-ackwards hierarchy **replicates** across random split-halves of the sample – Everett's (1983) factor comparability coefficients, reviving the split-half replication gate of the research program that produced the bass-ackwards method: Saucier (1997) screened factor solutions by their split-half stability, and Saucier, Georgiades, Tsaousis, and Goldberg (2005) chose the optimal hierarchical level by requiring split-half replication above a .90 threshold. It is also the direct instrument for the overextraction caution in `suggest_k()`: non-replicable structure concentrates in the deeper levels of an overextracted hierarchy (Forbes, 2023).

## Usage

```
comparability(
  data,
  k_max,
  engine = "pca",
  cor = "pearson",
  fm = "minres",
  n_splits = 10L,
  seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame or numeric matrix of observed variables (items in columns, observations in rows). **Raw data only** – splitting needs rows, so a correlation matrix is not accepted. Missing values are handled pairwise throughout (as in `ackwards()`'s default). |
| `k_max` | Maximum number of factors/components to evaluate – normally the same value (or one or two above it) you intend to pass to `ackwards()`. Required. |
| `engine` | Extraction engine: `"pca"` (default) or `"efa"`. `"esem"` is not yet supported here (fitting `2 * n_splits` lavaan hierarchies needs its own performance treatment); for ESEM workflows, run `comparability()` with `engine = "efa"` as a structural screen. |

| | |
|---|---|
| cor | Correlation basis: "pearson" (default) or "spearman". As with suggest_k(), "polychoric" is not supported (estimating polychoric matrices in every half-sample is slow and unstable); users analysing ordinal data should screen replicability on the Pearson basis and fit the final model with cor = "polychoric" in ackwards(). |
| fm | Factor extraction method passed to psych::fa(); only used when engine = "efa". One of "minres" (default), "ml", or "pa". |
| n_splits | Number of random split-half replicates. Default 10L. The published precedents used a single split (Saucier et al., 2005); repeating the split guards against the luck of one draw and is this implementation's robustness choice. Each replicate fits 2 * k_max solutions, so the default costs 20 hierarchy fits; PCA and EFA are fast enough that this is typically a few seconds. |
| seed | Integer seed for reproducible splits. NULL (default) uses the current RNG state. |
| ... | Reserved for future arguments. |

## Details

For each of n_splits random half-splits, solutions at every level 1..k_max are fit independently in each half. Each half-solution's factors are matched to the **full-sample** solution's factors (so coefficients are reported under the same m{k}f{j} labels you get from ackwards()), and the comparability coefficient for a factor is the correlation between its two matched half-solution scores, computed on the pooled correlation matrix via the same W'RW algebra used for between-level edges – applying both halves' scoring weights to the full sample, exactly Everett's procedure. Tucker's congruence coefficient (phi) between the matched half-solution loading columns is reported alongside: comparability asks whether the two halves' *scores* agree; phi asks whether their *loading patterns* agree.

## Value

An object of class "comparability". Print it for a per-level summary; call autoplot() on it for a diagnostic plot. The list contains:

| | |
|---|---|
| coefficients | Data frame with one row per split x level x factor: split, level, factor (full-sample m{k}f{j} label), r (score comparability), phi (Tucker's congruence of the matched loading columns). NA when the level did not converge in one of the halves. |
| summary | Data frame with one row per level x factor: level, factor, r_median, r_min, phi_median, phi_min (across splits), and n_splits_ok (splits in which both halves converged). |
| k_max | Deepest level evaluated. Can be lower than the k_max you asked for when the full-sample fit truncated (non-convergence at deep levels); the original request is kept in k_requested. |
| k_requested, n_splits, n_half, engine, cor, fm, n_obs, n_vars, seed | Metadata. |

## Interpreting the output

Coefficients near 1 mean the factor re-emerges in independent half-samples; a factor whose comparability is low is sample-idiosyncratic and should not anchor substantive interpretation. Conven-

tional benchmarks: **.90** is the split-half replication threshold of Goldberg's lexical research program (Saucier et al., 2005) and follows from Everett's (1983) rationale (split-half factors sharing at least 81% of their variance); **.95** is the stricter bound at which two factors are conventionally treated as interchangeable (Lorenzo-Seva & ten Berge, 2006). These are conventions, not tests, so `comparability()` reports every coefficient and flags nothing. The deepest level at which all factors replicate is a natural **hierarchy floor** for `ackwards()`'s `k_max`; see `vignette("ackwards-girard")` for the full workflow.

A level that fails to converge in a half-sample yields `NA` coefficients for that split (convergence is data, not an error); the number of usable splits per factor is reported in `summary$n_splits_ok` and a message summarises any shortfall.

## References

Everett, J. E. (1983). Factor comparability as a means of determining the number of factors and their rotation. *Multivariate Behavioral Research*, 18(2), 197–218. doi:10.1207/s15327906mbr1802_5

Saucier, G. (1997). Effects of variable selection on the factor structure of person descriptors. *Journal of Personality and Social Psychology*, 73(6), 1296–1312. doi:10.1037/00223514.73.6.1296

Saucier, G., Georgiades, S., Tsaousis, I., & Goldberg, L. R. (2005). The factor structure of Greek personality adjectives. *Journal of Personality and Social Psychology*, 88(5), 856–875. doi:10.1037/00223514.88.5.856

Lorenzo-Seva, U., & ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2(2), 57–64. doi:10.1027/16142241.2.2.57

Forbes, M. K. (2023). Improving hierarchical models of individual differences: An extension of Goldberg's bass-ackward method. *Psychological Methods*. doi:10.1037/met0000546

## See Also

`suggest_k()` for the plausible depth *range* (eigenstructure), `factorability()` for sampling adequacy before you fit, `prune()` for factors that perpetuate without differentiating (redundancy), and `ackwards()` for the extraction itself.

## Examples

```
cmp <- comparability(sim16, k_max = 5, n_splits = 5, seed = 1)
cmp
cmp$summary
```

---

`factor_labels` *Read the factor labels stored on an ackwards object*

---

## Description

Read the factor labels stored on an ackwards object

### Usage

```
factor_labels(x)
```

### Arguments

| | |
|---|---|
| x | An ackwards object. |

### Value

The named character vector of factor labels (names are factor IDs), or NULL if none have been set. See set_factor_labels() to attach them.

### See Also

set_factor_labels()

### Examples

```
x <- ackwards(sim16, k_max = 4, engine = "pca")
factor_labels(x) # NULL -- none set yet
x <- set_factor_labels(x, c(m4f1 = "Alpha"))
factor_labels(x)
```

---

## factorability *Screen a dataset for factorability and sampling adequacy*

---

### Description

Before extracting a hierarchy, it is worth asking two prior questions: *is this correlation matrix even worth factoring*, and *is the sample large enough to trust the answer*. factorability() reports the standard diagnostics for both, one object you can print or pull values from:

- **KMO** – the Kaiser-Meyer-Olkin measure of sampling adequacy, overall and per variable (via psych::KMO()). It contrasts correlations with partial correlations: a low value means the variables share little common variance once other variables are partialled out, so a factor model has little to recover.
- **Bartlett's test of sphericity** – tests whether the correlation matrix differs from the identity (via psych::cortest.bartlett()). Needs N; a non-significant result says the correlations are too weak to factor.
- **Sample size** – N, the number of variables p, and the N:p ratio.
- **Ledermann bound** – the largest number of *common factors* identifiable from p variables (a hard limit for EFA/ESEM, though not for PCA).

**Read these as conventions, not verdicts.** Every cutoff here – Kaiser's KMO bands, the 5:1 / 10:1 `N:p` rules of thumb, Bartlett significance at `.05` – is a widely repeated *rule of thumb*, not a settled threshold, and each is contested in the methodological literature (the required `N` in particular depends on communalities and factor overdetermination far more than on any fixed ratio; MacCallum et al. 1999). `factorability()` deliberately reports the numbers and their conventional bands rather than returning a pass/fail flag. `ackwards()` runs the same screen internally and warns only at the genuinely consequential extreme (KMO < `0.5`, `N:p` < 5).

## Usage

```
factorability(data, cor = c("pearson", "spearman", "polychoric"), n_obs = NULL)
```

## Arguments

| | |
|---|---|
| `data` | A data frame or numeric matrix of item responses, **or** a correlation matrix (detected automatically). KMO and Bartlett are computed on the correlation matrix in the basis you name via `cor`. |
| `cor` | The correlation basis to screen on: `"pearson"` (default), `"spearman"`, or `"polychoric"`. Ignored when `data` is already a correlation matrix. Use the same basis you plan to pass to `ackwards()`. |
| `n_obs` | Number of observations. Taken from `nrow(data)` for raw data (a supplied value is ignored with a warning). **Required** for the N-based diagnostics (Bartlett, `N:p`) when `data` is a correlation matrix; those are reported as `NA` when it is absent. |

## Value

An object of class `factorability` (a named list): `cor`, `n_obs`, `n_vars`, `np_ratio`, `kmo_overall`, `kmo_items` (a data frame of per-item MSA), `bartlett` (a list of `chisq`/`df`/`p_value`, or `NULL` when `N` is unknown), and `ledermann` (the bound). Print it for a banded summary; index the list for the raw values.

## References

Kaiser, H. F. (1974). An index of factorial simplicity. *Psychometrika*, 39(1), 31–36.

MacCallum, R. C., Widaman, K. F., Zhang, S., & Hong, S. (1999). Sample size in factor analysis. *Psychological Methods*, 4(1), 84–99.

## See Also

`check_items()` (per-item screening), `suggest_k()` (how many factors), and `comparability()` (split-half replicability) – the other pre-analysis diagnostics; `ackwards()`, which runs this screen internally.

## Examples

```
# Continuous data, Pearson basis
factorability(sim16)

# From a correlation matrix: supply n_obs for the N-based diagnostics
```

```
R <- cor(sim16)
factorability(R, n_obs = nrow(sim16))
```

---

`forbes2023` *Assessing Mental Health symptom correlation matrix (Forbes 2023 applied example)*

---

## Description

The 155 x 155 Spearman correlation matrix among 155 mental-health symptom variables that forms the applied example in Forbes (2023). It is a real, deep hierarchy: `ackwards(forbes2023, k_max = 10)` unfolds a general factor of psychopathology at the top down to 10 fine-grained components, the worked example that motivates the Forbes extension (`pairs = "all"`, `prune()`).

## Usage

```
forbes2023
```

## Format

A 155 x 155 numeric matrix of Spearman correlations: symmetric, unit diagonal, correlations in roughly `0.01`–`0.94`. Row and column names are the 155 symptom-variable labels (e.g. `Impulsivity`, `Blurting`).

## Details

Where `sim16` and `bfi25` are teaching foils, `forbes2023` is a fidelity/reproduction dataset: a large, messy, published case bundled so the package can reproduce the exact applied example analyzed in Forbes's paper.

The correlations come from the Assessing Mental Health (AMH) study – the Australian general-population sample (N = 3,175) of Forbes et al. (2021) – spanning symptoms of 18 DSM disorders. Being a correlation matrix, it carries no per-variable sample size; supply `n_obs = 3175` to `ackwards()` if you want EFA/ESEM fit statistics scaled to the original sample.

This matrix reproduces Forbes's published results exactly: the package regression test `test-forbes-fidelity.R` runs `ackwards()` on this exported `forbes2023` and matches her reference implementation's between-level correlations to `1.3e-14` across all 45 level-pairs at `k_max = 10`.

To regenerate this dataset, run `source("data-raw/forbes2023.R")` from the package root (downloads the source CSV from OSF).

## Source

Forbes's OSF project for the 2023 paper (file `corSpearman_AMH.csv`), https://osf.io/pcwm8/, redistributed here under its Creative Commons Attribution 4.0 International (CC-BY 4.0) license (see the package's `LICENSE.note`). The underlying data are from the Assessing Mental Health study (Forbes et al., 2021).

## References

Forbes, M. K. (2023). Improving hierarchical models of individual differences: An extension of Goldberg's bass-ackward method. *Psychological Methods*. doi:10.1037/met0000546

Forbes, M. K., Sunderland, M., Rapee, R. M., Batterham, P. J., Calear, A. L., Carragher, N., Ruggero, C., Zimmerman, M., Baillie, A. J., Lynch, S. J., Mewton, L., Slade, T., & Krueger, R. F. (2021). A detailed hierarchical model of psychopathology: From individual symptoms up to the general factor of psychopathology. *Clinical Psychological Science*, 9(2), 139–168. doi:10.1177/2167702620954799

## See Also

`vignette("ackwards-forbes2023")` for a full reproduction of the Forbes (2023) applied example on this dataset, and `vignette("ackwards-forbes")` for the method extension itself.

## Examples

```
dim(forbes2023)
forbes2023[1:3, 1:3]

# The Forbes (2023) applied example: a 10-level hierarchy from 155 symptoms.
x <- ackwards(forbes2023, k_max = 10, pairs = "all")
x
```

---

`glance.ackwards` *Glance at an ackwards object*

---

## Description

Returns a one-row data frame of top-level model metadata. For EFA and ESEM objects, fit indices at the deepest converged level are included. The same five columns (`CFI`, `TLI`, `RMSEA`, `SRMR`, `BIC`) are present across all engines; columns unavailable for a given engine or estimator are `NA` (e.g., `CFI` and `SRMR` are `NA` for EFA; all five are `NA` for PCA; for ESEM, `BIC` is `NA` under `estimator = "WLSMV"/"ULSMV"` – these limited-information estimators have no proper log-likelihood – and populated under `"ML"/"MLR"`). Under a scaled-test estimator (`"WLSMV"/"ULSMV"/"MLR"`) the `CFI`/`TLI`/`RMSEA` reported here are the scaled variants (see `tidy.ackwards()` for the rationale).

## Usage

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

## Arguments

| | |
|---|---|
| `x` | An `ackwards` object. |
| `...` | Ignored. |

### Value

A one-row `data.frame`.

### See Also

`tidy.ackwards()`, `print.ackwards()`

### Examples

```
x <- ackwards(sim16, k_max = 5)
glance(x)
```

---

## label_template *Generate a node-label scaffold for autoplot*

---

### Description

Returns a named character vector covering all factor IDs in the object, ready to pass to `autoplot(x, node_labels = ...)`. Printing the result shows an editable `c(...)` literal so you can copy it into a script, fill in substantive labels, and use it directly; assigning the result (`labs <- label_template(x)`) produces no console output.

### Usage

```
label_template(x, style = c("id", "forbes", "blank"))

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

### Arguments

| | |
|---|---|
| `x` | An `ackwards` object. |
| `style` | One of `"id"` (default), `"forbes"`, or `"blank"`. See **Style options** above. |
| `...` | Ignored; included for S3 method consistency. |

### Details

The factor IDs are returned in the same left-to-right, top-to-bottom order used by `ba_layout()` and `autoplot.ackwards()`, so the printed literal maps directly onto the diagram.

### Value

A named character vector of class `"ackwards_labels"`: names are factor IDs (`"m{k}f{j}"`), values are the label strings for the chosen `style`. The vector is suitable for direct use as the `node_labels` argument to `autoplot.ackwards()`. Its `print()` method renders the vector as an editable `c(...)` literal for copy-paste.

### Style options

- `"id"` *(default)* – every value equals the factor ID (`"m1f1"`, `"m2f1"`, ...). This is a round-trip no-op: passing the result to `node_labels` without editing reproduces the default labels exactly. Useful as the starting point for adding substantive labels.
- `"forbes"` – values follow the Forbes (2023) convention: level-letter + within-level index (`"A1"`, `"B1"`, `"B2"`, ...). Level 1 -> `A`, level 2 -> `B`, level 3 -> `C`, and so on. Within-level indices are assigned in canonical layout order (left to right). Requires `k_max <= 26` (LETTERS has 26 entries); an error is raised for deeper objects.
- `"blank"` – all values are empty strings. Useful as a starting scaffold when you want to supply every label from scratch with no defaults showing through.

### See Also

`autoplot.ackwards()`, `top_items()`, `ba_layout()`

### Examples

```
x <- ackwards(sim16, k_max = 5)

# Start from ID defaults, then fill in your own labels:
labs <- label_template(x)
labs["m5f1"] <- "My factor name"

# Forbes letter convention:
label_template(x, style = "forbes")
```

---

## predict.ackwards *Score new observations with a fitted ackwards model*

---

### Description

Applies a fitted bass-ackwards model to new data — for example the held-out test split of a cross-validation design — producing factor scores for every level **without retraining**. This is the idiomatic `predict()` front door to the same machinery as `augment.ackwards()`: the call `predict(object, newdata)` returns exactly `augment(object, data = newdata, append = FALSE)`, a data frame holding only the `.m{k}f{j}` score columns, one row per row of `newdata`.

### Usage

```
## S3 method for class 'ackwards'
predict(object, newdata, scaling = c("fit", "sample"), ...)
```

## Arguments

| | |
|---|---|
| `object` | An `ackwards` object. |
| `newdata` | A data frame or numeric matrix with the same variables (columns) used to fit `object`. Required — to retrieve scores stored at fit time, use `augment(object)` instead. |
| `scaling` | Which item means/SDs standardize `newdata`: `"fit"` (default, the training moments) or `"sample"` (`newdata`'s own moments). See `augment.ackwards()`. |
| `...` | Ignored. |

## Details

Under the default `scaling = "fit"`, `newdata` is standardized by the **fit-time** item means/SDs stored in the object before the stored weight matrices are applied, so the new scores land on the same metric as the training solution: an observation's score does not depend on which other observations share its split, and train and test scores are directly comparable. See `augment.ackwards()` (section *Scoring new observations*) for the full semantics, the `scaling = "sample"` alternative, and the non-Pearson-basis caveat.

`newdata` must contain the variables the model was fit on (matched by column name, with extra columns ignored; a bare unnamed matrix is matched positionally). Rows with missing items produce `NA` scores (scoring does not impute).

## Value

A data frame of factor scores (columns `.m{k}f{j}`, one per factor per level), with one row per row of `newdata` in the original order.

## See Also

`augment.ackwards()` for appending scores to the data (and the full scoring documentation), `ackwards()`.

## Examples

```
# Cross-validation: fit on a training split, score the test split
train <- sim16[1:500, ]
test <- sim16[501:1000, ]
x <- ackwards(train, k_max = 5)
test_scores <- predict(x, test)
head(test_scores)

# Identical to the augment() spelling:
identical(test_scores, augment(x, data = test, append = FALSE))
```

## print.ackwards *Print an ackwards object*

### Description

Displays a compact summary of the bass-ackwards result using cli formatting. No matrix dumps – use `tidy.ackwards()` to access values programmatically.

### Usage

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

### Arguments

| | |
|---|---|
| x | An ackwards object. |
| ... | Ignored. |

### Value

x invisibly.

### See Also

`tidy.ackwards()`, `glance.ackwards()`

## print.check_items *Print an item quality check*

### Description

Print an item quality check

### Usage

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

### Arguments

| | |
|---|---|
| x | A check_items object (produced by `check_items()`). |
| ... | Ignored. |

### Value

x invisibly.

## print.comparability *Print a comparability object*

### Description

Print a comparability object

### Usage

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

### Arguments

| | |
|---|---|
| x | A comparability object. |
| ... | Ignored. |

### Value

x invisibly.

## print.factorability *Print a factorability screen*

### Description

Print a factorability screen

### Usage

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

### Arguments

| | |
|---|---|
| x | A factorability object (produced by factorability()). |
| ... | Ignored. |

### Value

x invisibly.

## print.suggest_k *Print a suggest_k object*

### Description

Print a suggest_k object

### Usage

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

### Arguments

| | |
|---|---|
| x | A suggest_k object. |
| ... | Ignored. |

### Value

x invisibly.

## print.summary_ackwards *Print a summary_ackwards object*

### Description

Print a summary_ackwards object

### Usage

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

### Arguments

| | |
|---|---|
| x | A summary_ackwards object (produced by summary()). |
| ... | Ignored. |

### Value

x invisibly.

## print.top_items *Print a top_items object*

### Description

Print a top_items object

### Usage

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

### Arguments

| | |
|---|---|
| x | A top_items object (produced by top_items()). |
| ... | Ignored. |

### Value

x invisibly.

## prune *Flag redundant or artifactual factors (Forbes 2023 extension)*

### Description

prune() never removes anything from an ackwards object – it only annotates factors with pruned/prune_reason flags in x$prune$nodes (flag-only, never removes; the object keeps every level). Because it is a separate, cheap step from extraction, you can re-prune with new thresholds without re-running ackwards():

```
x <- ackwards(bfi25, k_max = 6, engine = "esem")  # expensive
x |> prune("redundant")                            # cheap, repeatable
x |> prune("redundant", redundancy_r = 0.95)       # no re-extraction
```

prune() is an S3 generic (rather than a plain function) so it coexists with the prune generics already defined by recursive-partitioning packages (e.g. rpart::prune) regardless of package load order.

## Usage

```
prune(x, ...)

## S3 method for class 'ackwards'
prune(
  x,
  rules = "none",
  manual = NULL,
  redundancy_r = 0.9,
  redundancy_phi = NULL,
  redundancy_criterion = c("direct", "adjacent"),
  min_items = 3L,
  orphan_r = 0.5,
  ...
)
```

## Arguments

| | |
|---|---|
| x | An ackwards object. |
| ... | Reserved for future methods/arguments. |
| rules | Character vector controlling which auto-rules run. Default "none" (no auto rule; combine with manual for pure manual pruning, or call prune(x) with no arguments to clear any existing pruning). Options:<br>• "redundant" – identify chains of factors connected by score correlations \|r\| >= redundancy_r (and optionally phi > redundancy_phi), using redundancy_criterion (default "direct", faithful to Forbes). Applies Forbes's (2023) retention rule: keep the bottom node when the chain reaches level k_max (most specific); keep the top node otherwise. Flagged nodes get pruned = TRUE and prune_reason = "redundant" in x$prune$nodes.<br>• "artifact" (or the alias "artefact", normalized to "artifact") – compute Tucker's congruence coefficient (phi) for all cross-level factor pairs and store in x$prune$phi, plus structural signals (few_items/orphan/split_merge) in x$prune$structural. No factors are auto-flagged; artifact identification requires judgment (Forbes, 2023; Wicherts et al., 2016). |
| manual | Character vector of factor labels (e.g. c("m4f3", "m4f4")) to flag directly, in addition to or instead of an auto rule. Standalone manual pruning is supported: prune(x, manual = c("m4f3")). Unknown labels error. A node already flagged by an auto rule keeps that prune_reason; only otherwise-unflagged manual nodes get prune_reason = "manual". |
| redundancy_r | Scalar in (0, 1]. Score-correlation \|r\| threshold for redundancy chains. Default 0.9 (Forbes, 2023). |
| redundancy_phi | Scalar in (0, 1], NULL (default, auto), or NA (explicit opt-out). When NULL:<br>• x$engine == "pca" – no phi filter. Component scores are *determinate* (exact linear functions of the data, with no factor-score indeterminacy), so the score correlation \|r\| is the true correlation between the components themselves; phi adds nothing that \|r\| does not already capture. |

- `x$engine` is `"efa"` or `"esem"` – automatically set to `0.95` (Lorenzo-Seva & ten Berge, 2006). Factor-score indeterminacy off-PCA means `|r|`-alone is liberal; the conjunctive phi criterion is the conservative default. A cli message announces the resolved value. Pass `NA` to disable phi filtering regardless of engine. Pass a numeric value to override on any engine.

`redundancy_criterion`
: How redundancy chains are traced. One of:

  - `"direct"` (default) – chase upward via the **direct (skip-level)** correlation between a factor and each ancestor level, continuing while `|r| >= redundancy_r` contiguously. This is Forbes's (2023) published `ChaseCorrPaths` rule and the honest operationalization of "the same construct" (two factor scores share `>= redundancy_r^2` variance *directly*). It reproduces her AMH applied example exactly.
  - `"adjacent"` – trace **adjacent primary-parent** links only (each consecutive level `|r| >= redundancy_r`). This was the pre-M53 default; because correlation is non-transitive it can both over- and under-flag versus `"direct"` in deep (many-level) hierarchies, so it is retained only as an opt-in. On shallow/transitive hierarchies the two agree.

`min_items`
: Minimum number of items for which a factor must be the primary loader (highest `|loading|`). Factors with fewer than `min_items` primary items are flagged `few_items = TRUE` in `x$prune$structural`. Only used when `rules` includes `"artifact"`. Default `3L` – a factor defined by one or two items is under-identified and frequently an extraction artifact rather than a replicable construct (the classic "three-indicator rule"; Forbes, 2023, Fig. 2).

`orphan_r`
: Threshold for the `orphan` structural signal. A factor whose maximum **adjacent-level** `|r|` (to the immediately shallower and deeper levels) falls below `orphan_r` is flagged `orphan = TRUE` in `x$prune$structural` – it does not connect to the neighbouring solutions and so does not replicate across the hierarchy. Only used when `rules` includes `"artifact"`. Default `0.5` – a moderate correlation; a factor that shares less than a quarter of its variance with every neighbour is a structural outlier worth inspecting.

## Details

**Reading `x$prune$chains` under `redundancy_criterion = "direct"`.** The `r_to_prev` and `phi_to_prev` columns report the **adjacent-level** correlation and congruence between consecutive chain members (for continuity of display), but chain *membership* is decided by the **direct (skip-level)** correlation to the chain's deepest factor. A direct chain can therefore legitimately contain a link whose `r_to_prev` is *below* `redundancy_r` – the stronger direct link is what justified it. The `endpoint_r` column gives the direct root-to-leaf correlation as an at-a-glance cross-check. Under `redundancy_criterion = "adjacent"`, `r_to_prev` *is* the criterion and always meets `redundancy_r`.

## Value

`x`, with `$prune` populated (replacing any prior pruning).

## References

Forbes, M. K. (2023). Improving hierarchical models of individual differences: An extension of Goldberg's bass-ackward method. *Psychological Methods*. doi:10.1037/met0000546

Lorenzo-Seva, U., & ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2(2), 57–64. doi:10.1027/16142241.2.2.57

## See Also

`ackwards()`, `tidy.ackwards()` (`what = "nodes"`), `autoplot.ackwards()` (`drop_pruned`)

## Examples

```
# sim16 has a planted redundant chain + overextraction artifact at k = 5,
# so the prune rules always have a finding to show (and no ordinal warning).
x <- ackwards(sim16, k_max = 5)

xp <- prune(x, "redundant")
xp$prune$nodes

# Re-prune with a new threshold -- no re-extraction needed
prune(x, "redundant", redundancy_r = 0.95)

# Manual pruning: standalone, or mixed with an auto rule
prune(x, manual = "m4f2")
prune(x, "redundant", manual = "m4f2")
```

---

# set_factor_labels *Attach persistent factor labels to an ackwards object*

---

## Description

Store substantive names for factors (e.g. `"Neuroticism"` for `"m5f1"`) on the object itself, so that print(), summary(), tidy(), autoplot(), and `top_items()` display them without re-supplying the labels each time. This is the *factor*-label counterpart to item / variable labels (see `top_items()` and `?ackwards`); the two are distinct and never interchanged.

## Usage

```
set_factor_labels(x, labels)
```

## Arguments

| | |
|---|---|
| `x` | An `ackwards` object. |
| `labels` | A named character vector mapping factor IDs (`"m{k}f{j}"`) to label strings, or `NULL` to clear all stored labels. Every name must match a factor ID in `x` (an unknown ID is an error, not a warning – the object's IDs are knowable up front; see `factor_labels()` to read the current set). |

### Details

Labels are display only – they never change a factor's stable ID (`m{k}f{j}`), and every lineage / edge / score column continues to key on the ID. A factor with no label falls back to its ID everywhere. The stored labels ride along through `prune()`, `boot_edges()`, `augment()`, and `predict()` unchanged.

### Value

The `ackwards` object, with `meta$factor_labels` updated. Pipeable.

### Updating and clearing

`set_factor_labels()` **merges** into any labels already stored, so you can build them up incrementally. Within a single call:

- a normal string sets (or overwrites) that factor's label;
- an `NA` or `""` value **removes** just that factor's label;
- passing `labels = NULL` clears **all** labels at once.

The scaffold printed by `label_template()` is a convenient starting point: copy its `c(...)` literal, fill in the substantive names, and pass it here.

### See Also

`factor_labels()` to read them back, `label_template()` for a ready-to-edit scaffold, `autoplot.ackwards()` (`node_labels` overrides a stored label per node).

### Examples

```
x <- ackwards(sim16, k_max = 4, engine = "pca")

# Start from a scaffold, fill in names, attach them:
x <- set_factor_labels(x, c(m4f1 = "Alpha", m4f2 = "Beta"))
factor_labels(x)

# Merge in more; remove one; everything downstream now shows the labels:
x <- set_factor_labels(x, c(m2f1 = "Broad", m4f2 = NA))
summary(x)
```

---

## sim16 — *Simulated continuous bass-ackwards teaching example (16 items, known hierarchy)*

---

### Description

A 1 000-row, fully continuous dataset simulated from a population model with a known 1 -> 2 -> 4 bass-ackwards hierarchy, for showcasing the default `cor = "pearson"` extraction path without the ordinal-detection warning that `bfi25`'s Likert items trigger.

## Usage

```
sim16
```

## Format

A data frame with 1 000 rows and 16 numeric columns (i1–i16, continuous, no missing values).

## Details

**Population model.** `Sigma = Lambda %*% Phi %*% t(Lambda) + Psi`, an oblique common-factor model with 4 true group factors, sampled via a Cholesky factorization (base R; no `MASS` dependency):

| Factor | Items | Metatrait |
|---|---|---|
| f1 | i1–i4 | 1 (with f2) |
| f2 | i5–i8 | 1 (with f1) |
| f3 | i9–i12 | 2 (with f4) |
| f4 | i13–i16 | 2 (with f3) |

All items load `0.75` on their true factor (no cross-loadings). Factor correlations: `0.45` within a metatrait (f1-f2, f3-f4), `0.15` between metatraits. Uniquenesses are `1 - communality` (uniform `0.4375` by the symmetry of the design above).

**Ground-truth hierarchy** (verified against `ackwards(engine = "efa")`): `k=1` recovers a single general factor across all 16 items; `k=2` splits along the metatrait line (i1-i8 vs. i9-i16); `k=4` recovers the 4 true group factors exactly; all six `suggest_k()` recommendations (five criteria – VSS reports at complexities 1 and 2) reach a consensus of `k = 4`.

**Idealized by design.** The planted signal is strong and clean, so all six `suggest_k()` recommendations converge on `k = 4` – deliberately the *easy* case, for building intuition about what recovering a known hierarchy looks like. Real data rarely agree this cleanly: on `bfi25` the same criteria span `k = 4–6`. The two datasets are complementary teaching foils – `sim16` for "watch the method recover a structure we planted," `bfi25` for "reason about a hierarchy when the criteria disagree." Present `sim16`'s consensus as the ideal, not the norm.

**Deliberate overextraction artifact at k=5.** The population has exactly 4 factors, so requesting a 5th finds no real dimension: EFA produces an orphan factor with zero primary-loading items. With `prune(x, "artifact")` (default `min_items = 3`, `orphan_r = 0.5`), that factor is flagged both `few_items` and `orphan`. Because the true (non- splitting) factors persist essentially unchanged from `k=3` onward, their parent-child score correlations approach 1 and are flagged by `prune(x, "redundant")` (`|r| >= .9` and, under the EFA auto-default, Tucker's phi `> .95`). This is a textbook overextraction artifact, included so the Forbes/redundancy examples have a guaranteed finding to teach against (unlike `bfi25`, which does not reliably trigger one).

To regenerate this dataset, run `source("data-raw/sim16.R")` from the package root (`set.seed(42)`).

## Source

Simulated; see `data-raw/sim16.R` for the full generative model.

## Examples

```
dim(sim16)
head(sim16)
```

---

`suggest_k` *Suggest a maximum number of factors for bass-ackwards analysis*

---

## Description

Runs complementary factor-retention criteria and reports their recommendations. No single criterion is definitive; the goal is a consensus range to inform your choice of `k` in `ackwards()`.

## Usage

```
suggest_k(
  data,
  k_max = NULL,
  criteria = c("pa_pc", "pa_fa", "map", "vss", "cd"),
  cor = "pearson",
  n_obs = NULL,
  n_iter = 20L,
  seed = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame or numeric matrix (items in columns, observations in rows). Alternatively, a pre-computed **correlation matrix** may be supplied (a square, symmetric, numeric matrix with unit diagonal). When a correlation matrix is supplied, `n_obs` is required (PA and VSS need N), the `cor` argument is ignored, and the Comparison Data (CD) criterion is skipped (CD requires raw item distributions for resampling). |
| `k_max` | Maximum number of factors/components to *evaluate* when recommending a depth – not a depth itself. Defaults to `min(ncol(data) - 1, 8)`. Increase if you expect a deeper hierarchy. (Note: this is a distinct meaning from `ackwards()`'s `k_max`, which *is* the extraction depth; the two share a name because they're the same dial in the `suggest_k() -> ackwards()` workflow, just applied at different stages.) |
| `criteria` | Character vector of criteria to compute. Any subset of `c("pa_pc", "pa_fa", "map", "vss", "cd")`; default is all five. `"pa_pc"` and `"pa_fa"` share one parallel-analysis call (both or neither are fast); `"map"` and `"vss"` share one VSS call. Criteria not requested are skipped entirely (no computation) and their `k_*` fields in the result are `NA`. `"vss"` selects both VSS-1 and VSS-2 as a unit. |
| `cor` | Correlation basis: `"pearson"` (default) or `"spearman"`. Should match the `cor` argument you plan to use in `ackwards()`. Ignored when `data` is a correlation matrix (the basis is already fixed). |

| | |
|---|---|
| n_obs | Number of observations. Required when data is a pre-computed correlation matrix (PA and VSS need N). Ignored when raw data are supplied (N is determined from nrow(data)). |
| n_iter | Number of Monte Carlo iterations for parallel analysis. Default 20. Reduce to 5 for fast/exploratory runs; increase to 100+ for publication. |
| seed | Integer seed passed to set.seed() before the Comparison Data (CD) step. NULL (default) uses the current RNG state. **Note:** the parallel-analysis step uses psych::fa.parallel(), which does not respond reliably to set.seed() – PA simulation results will vary across calls regardless of seed. |
| ... | Reserved for future arguments. |

## Details

**Criteria available (controlled by the** criteria **argument):**

- **PA-PC** ("pa_pc") – Horn (1965) parallel analysis on PC eigenvalues. Compares observed eigenvalues to those from random correlation matrices; suggests retaining components whose eigenvalues exceed the 95th percentile of chance. Tends to overextract; treat as an upper bound.
- **PA-FA** ("pa_fa") – Horn (1965) parallel analysis using common-factor eigenvalues. More conservative than PA-PC and the better match for the EFA and ESEM engines in ackwards(). Shares one psych::fa.parallel() call with PA-PC.
- **MAP** ("map") – Velicer (1976) Minimum Average Partial criterion. Finds the k that minimises the average squared partial correlation remaining after extracting k components. Usually conservative. Shares one psych::vss() call with VSS.
- **VSS** ("vss") – Revelle & Rocklin (1979) Very Simple Structure fit at complexities 1 and 2 (VSS-1 and VSS-2). Finds the k maximising the fit of a very simple loading structure. Shares one psych::vss() call with MAP.
- **CD** ("cd") – Ruscio & Roche (2012) Comparison Data. Resamples from the observed item distributions to generate comparison eigenvalue profiles; retains factors until adding one no longer improves RMSE beyond chance. Requires the **EFAtools** package (install separately). Skipped with an informational message when **EFAtools** is absent or when a correlation matrix is supplied.

PA (both bases), MAP, and VSS share the same correlation matrix as ackwards(). CD operates on the raw data matrix directly (required for resampling).

## Value

An object of class "suggest_k". Print it for a formatted summary; call autoplot() on it for a diagnostic scree/criteria plot. The list contains:

| | |
|---|---|
| k_parallel_pc | Recommended k from PC-based parallel analysis (NA_integer_ when "pa_pc" not in criteria). |
| k_parallel_fa | Recommended k from FA-based parallel analysis (NA_integer_ if no FA factor exceeded the random threshold, or when "pa_fa" not in criteria). |
| k_map | Recommended k from MAP (NA_integer_ when "map" not in criteria). |

`k_vss1` Recommended k from VSS complexity-1 (`NA_integer_` when `"vss"` not in `criteria`).

`k_vss2` Recommended k from VSS complexity-2 (`NA_integer_` when `"vss"` not in `criteria`).

`k_cd` Recommended k from Comparison Data (`NA_integer_` when `"cd"` not in `criteria`, **EFAtools** is not installed, or CD fails).

`cd_available` Logical; `TRUE` when `"cd"` was requested, **EFAtools** was found, and CD ran successfully.

`criteria` Data frame with one row per k: `k`, `ev_obs` (observed PC eigenvalue), `ev_obs_fa` (observed FA eigenvalue), `pa_pc_quant` / `pa_fa_quant` (95th-pct simulated eigenvalue for each basis), `pa_pc_suggested` / `pa_fa_suggested` (logical retention), `map`, `vss1`, `vss2`. Columns for non-requested criteria are `NA`.

`criteria_requested` Character vector of the criteria that were requested (and therefore computed).

`k_max`, `n_obs`, `n_vars`, `cor` Metadata.

## Interpreting the output

k in `ackwards()` is a **maximum depth**, not a claim that exactly k factors exist. Users commonly set k one or two levels above the consensus to watch higher-level factors fragment – this is a feature of the method, not overextraction.

Treating retention estimates as a range rather than a verdict has direct support in the parallel-analysis literature: Lim and Jahng (2019) recommend interpreting the PA estimate as a range of roughly plus or minus one factor, resolved by interpretability, and Achim (2021) argues that even that overstates PA's precision – the disagreement itself is why `suggest_k()` reports several criteria and a consensus range, never a single number.

## Ordinal (Likert) data

`suggest_k()` screens on the Pearson or Spearman basis by design and never computes polychoric correlations itself, so when the raw data look ordinal (at most 7 distinct integer values per column) it emits a one-per-session warning – the same `detect_ordinal()` signal `ackwards()` and `comparability()` use (Invariant 6: announce consequential defaults loudly). Crucially, the advice points at the *final* `ackwards()` fit (`cor = "polychoric"`), **not** at `suggest_k()` itself: the screening basis is intentional. The plausible-k *range* is robust to the Pearson-vs-polychoric choice, so screening on Pearson and fitting the final model polychoric is the recommended workflow – computing polychoric correlations at the screening stage would only add cost and NPD risk without changing the range. The warning is skipped for correlation-matrix input (there are no items to inspect).

## A note on overextraction

PA-PC in particular tends to recommend more factors than replicate across independent samples, especially with correlated items (Forbes, 2023). This is a long-standing observation in practice: Saucier (1997, footnote 14) reported parallel analysis suggesting as many as 30 factors in wide lexical item sets. PA-FA and CD are more conservative. Treat the full set of criteria as a range: the true k is likely somewhere in the middle.

## References

Achim, A. (2021). Determining the number of factors using parallel analysis and its recent variants: Comment on Lim and Jahng (2019). *Psychological Methods*, 26(1), 69–73. doi:10.1037/met0000269

Forbes, M. K. (2023). Improving hierarchical models of individual differences: An extension of Goldberg's bass-ackward method. *Psychological Methods*. doi:10.1037/met0000546

Horn, J. L. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30, 179–185.

Lim, S., & Jahng, S. (2019). Determining the number of factors using parallel analysis and its recent variants. *Psychological Methods*, 24(4), 452–467. doi:10.1037/met0000230

Revelle, W., & Rocklin, T. (1979). Very simple structure: An alternative procedure for estimating the optimal number of interpretable factors. *Multivariate Behavioral Research*, 14(4), 403–414.

Ruscio, J., & Roche, B. (2012). Determining the number of factors to retain in an exploratory factor analysis using comparison data of a known factorial structure. *Psychological Assessment*, 24(2), 282–292.

Saucier, G. (1997). Effects of variable selection on the factor structure of person descriptors. *Journal of Personality and Social Psychology*, 73(6), 1296–1312. doi:10.1037/00223514.73.6.1296

Velicer, W. F. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika*, 41, 321–327.

## See Also

`ackwards()`; `factorability()`, `check_items()`, and `comparability()`, the other pre-analysis diagnostics.

## Examples

```
sk <- suggest_k(sim16)
sk
autoplot(sk)

# Run only MAP (fast; skips parallel analysis and CD)
suggest_k(sim16, criteria = "map")

# Run only the parallel-analysis criteria
suggest_k(sim16, criteria = c("pa_pc", "pa_fa"), n_iter = 5)

# Faster exploratory run
suggest_k(sim16, k_max = 6, n_iter = 5)

# Correlation-matrix input (CD is skipped; n_obs required)
R <- cor(sim16)
suggest_k(R, n_obs = nrow(sim16))
```

## summary.ackwards *Summarise an ackwards object*

### Description

Returns a structured summary_ackwards object that, when printed, shows per-level variance and fit indices, a readable lineage list, and (when present) pruning annotations. More verbose than print.ackwards(); designed for inspection and reporting.

### Usage

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

### Arguments

| | |
|---|---|
| object | An ackwards object. |
| ... | Ignored. |

### Value

An object of class "summary_ackwards", printed via print.summary_ackwards().

### See Also

print.ackwards(), tidy.ackwards(), glance.ackwards()

### Examples

```
x <- ackwards(sim16, k_max = 5)
summary(x)
```

## tidy.ackwards *Tidy an ackwards object into a long data frame*

### Description

Returns structured data from an ackwards object in tidy format. The default (what = "edges") returns the graph edge list that drives diagrams.

## Usage

```
## S3 method for class 'ackwards'
tidy(
  x,
  what = c("edges", "loadings", "variance", "fit", "nodes", "scores"),
  primary_only = FALSE,
  sort = c("none", "strength"),
  format = c("long", "wide"),
  conf_level = 0.95,
  ...
)
```

## Arguments

| | |
|---|---|
| x | An ackwards object. |
| what | What to extract: |

- "edges" *(default)* – one row per directed between-level edge: from, to, level_from, level_to, r, is_primary, above_cut. If boot_edges() has been run on the object, four bootstrap columns are appended: se, lo, hi (bootstrap standard error and percentile confidence-interval endpoints), and n_boot_ok (usable replicates).
- "loadings" – one row per item x factor x level: level, factor, item, loading, se, ci_lower, ci_upper. se, ci_lower, and ci_upper are populated only for engine = "esem" (which produces rotation-aware loading SEs); they are NA for PCA and EFA. The confidence level is controlled by conf_level.
- "variance" – one row per factor x level: level, factor, proportion, cumulative. Both are proportions of total item variance on a 0-1 scale (multiply by 100 for a percentage).
- "fit" – one row per fit statistic x level: level, statistic, value. For PCA objects the statistics are eigenvalues; for EFA objects they are chi, dof, p_value, RMSEA, TLI, BIC – where chi is the likelihood-ratio chi-square (psych::fa()'s STATISTIC), so chi, dof, p_value, RMSEA, and TLI all share one statistical framing (psych's residual-based *empirical* chi-square is a different statistic and is not reported); for ESEM they are chi, dof, p_value, CFI, TLI, RMSEA, SRMR, BIC. For ESEM under a scaled-test estimator ("WLSMV"/"ULSMV" for ordinal data, "MLR" for continuous), the whole row – chi/dof/ p_value **and** CFI/TLI/RMSEA – reports lavaan's mean-and-variance-adjusted ("scaled") variant, so every quantity shares one scaling. This matters most for WLSMV/ULSMV: the naive chi-square has no valid reference distribution (lavaan's own summary() labels its p-value "Unknown"), and the naive CFI/TLI are badly optimistic for ordinal data (Xia & Yang, 2019). "ML" has no scaled variant, so it reports the naive values (the correct ones for ML). SRMR has no scaled variant and is reported as-is. BIC is NA under WLSMV/ULSMV (no proper log-likelihood for a limited-information estimator) and populated under ML/MLR. Use format = "wide" for one row per **non-anchor** level (k >= 2; the saturated

1-factor anchor is dropped, matching `summary()` and `autoplot(what = "fit")`), one column per statistic. Conventional fit cutoffs (Hu & Bentler 1999) are shown as reference lines in `autoplot(what = "fit")` and inline in `summary()`, but are not returned as a pass/fail column here – they are contested thresholds, report-only, and never gate anything (see those functions' docs). `format` is oriented to the EFA/ESEM model-fit statistics; for PCA the "statistics" are per-component eigenvalues.

- `"nodes"` – Forbes-extension pruning annotations (requires `prune != "none"` when the object was created). One row per factor across all levels: `id`, `level`, `pruned`, `prune_reason`. Returns an empty data frame with the same columns when no pruning was applied.
- `"scores"` – long-format per-observation factor scores (requires `keep_scores = TRUE` at fit time or use `augment.ackwards()` for on-the-fly computation). Columns: `obs` (row index), `level`, `factor`, `score`.

`primary_only` For `what = "edges"` only. When `TRUE`, returns just each factor's primary-parent edge (`is_primary == TRUE`) – the lineage tree that the diagram draws as solid arrows. Default `FALSE` (all edges). Errors for any other value of `what`.

`sort` For `what = "edges"` only. One of `"none"` (default, natural order) or `"strength"` (descending `|r|`). Ignored for all other values of `what`.

`format` For `what = "fit"` only. One of `"long"` (default, one row per statistic x level) or `"wide"` (one row per level, one column per statistic). Errors for all other values of `what`.

`conf_level` For `what = "loadings"` only. Confidence level for the loading intervals; default `0.95`. The intervals are computed as `loading ± qnorm((1 + conf_level) / 2) * se` and are `NA` for engines that carry no SEs (PCA, EFA). Errors for all other values of `what`.

`...` Ignored.

## Value

A data frame (class `data.frame`).

## Factor labels

If factor labels have been attached to the object, the output gains display-only label columns: `factor_label` for `what = "loadings"`, `"variance"`, or `"scores"`, and `from_label`/`to_label` for `what = "edges"`. Each carries the label for a labeled factor and `NA` otherwise. These columns are **absent** when no labels are set, so an unlabeled object's output is unchanged; the ID columns (`factor`, `from`, `to`) are never altered.

## See Also

`glance.ackwards()`, `print.ackwards()`, `set_factor_labels()`

## Examples

```
x <- ackwards(sim16, k_max = 5)
tidy(x) # edges in natural order
```

```
tidy(x, sort = "strength") # strongest edges first
tidy(x, primary_only = TRUE) # just the primary-parent lineage
tidy(x, what = "loadings")
tidy(x, what = "variance")
tidy(x, what = "fit")
tidy(x, what = "fit", format = "wide")
```

---

`top_items` *Display the salient items for each factor*

---

## Description

Returns, per level and factor, the items whose absolute loading meets or exceeds `cut`, sorted by descending absolute loading. This gives a concise reading of "what each factor is about" without printing a full item-by-factor matrix, which does not scale well to large `k` or many items.

## Usage

```
top_items(
  x,
  level = NULL,
  cut = 0.3,
  n = NULL,
  sort = TRUE,
  by = c("factor", "item"),
  show_labels = TRUE
)
```

## Arguments

| | |
|---|---|
| `x` | An `ackwards` object. |
| `level` | Integer vector selecting which level(s) to include. `NULL` (default) returns all levels. |
| `cut` | Absolute-loading threshold. Items with `\|loading\| >= cut` are shown. Default `0.3`, matching the `cut_show` plotting default. |
| `n` | Maximum number of items to show per factor. `NULL` (default) shows all items meeting the cut. When set, the top-`n` items by `\|loading\|` are kept after applying the cut. |
| `sort` | Logical. When `TRUE` (default), items within each group are ordered by descending `\|loading\|`. Set to `FALSE` to keep the original order (useful when items have a meaningful sequence). |
| `by` | One of `"factor"` (default) or `"item"`. `"factor"` groups the listing by factor (the salient items *of* each factor – "what is this factor about?"). `"item"` inverts the grouping to list, for each item, the factors it loads on – which makes cross-loadings legible ("where does this item go?"). `n` and `sort` apply within whichever unit `by` selects. |

show_labels Logical. When TRUE (default) and the data carried a variable-label attribute at fit time (see ackwards()), items are shown as id: label; items without a label fall back to the bare id. Set to FALSE to always show the bare m{k}f{j}-style item ids.

## Details

Loadings are signed and reflect the object's primary-parent sign alignment (see ackwards()). Items near the cut threshold may appear for one sign orientation but not the other; this is expected and informative.

If factor labels have been attached, the factor dimension is shown as label (id) wherever it appears – the group headers under by = "factor" and the body entries under by = "item".

## Value

An object of class "top_items". Print it for a grouped cli listing. The underlying data frame (one row per selected item) is accessible via $data and contains columns level, factor, item, and loading (plus label when labels are available). The values equal the corresponding tidy(x, what = "loadings") rows (after filtering and optional sorting).

## See Also

tidy.ackwards(), label_template(), set_factor_labels(), autoplot.ackwards()

## Examples

```
# Fit the raw dataset (not na.omit(), which would drop the column
# attributes): bfi25's IPIP item labels are then captured and printed as
# `code: label`. `missing = "listwise"` handles the NAs cleanly. A 10-item
# subset keeps the example fast; use all items in practice.
x <- ackwards(bfi25[, 1:10], k_max = 3, cor = "polychoric", missing = "listwise")
top_items(x)
top_items(x, level = 3, cut = 0.4, n = 5)

# Invert the grouping to read cross-loadings item-by-item
top_items(x, level = 3, cut = 0.25, by = "item")
```

# Index