

Package ‘ackwards’

October 19, 2026

Title Bass-Ackwards Hierarchical Structural Analysis

Version 0.3.0

Description Implements Goldberg's (2006) <[doi:10.1016/j.jrp.2006.01.001](https://doi.org/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. It then describes the hierarchy through the correlations between the factor scores of different levels. It computes them with linear algebra from the scoring weights and the correlation matrix the fit uses, so they are exact for that matrix. Waller (2007) <[doi:10.1016/j.jrp.2006.08.005](https://doi.org/10.1016/j.jrp.2006.08.005)> showed, for principal components, that these correlations follow from the correlation matrix alone. Includes the Forbes (2023) <[doi:10.1037/met0000546](https://doi.org/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, GPArotation, 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, oblique, 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 8.1.0

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 CRAN

Date/Publication 2026-10-19 14:40:02 UTC

Contents

ackwards	3
augment.ackwards	10
autoplot	13
autoplot.ackwards	13
autoplot.comparability	19
autoplot.suggest_k	21
ba_layout	22
bfi25	23
boot_edges	24
check_items	27
comparability	28
factorability	31
factor_labels	33
forbes2023	33
glance.ackwards	35
label_template	36
predict.ackwards	37
print.ackwards	38
print.check_items	39
print.comparability	39
print.factorability	40
print.suggest_k	40
print.summary_ackwards	41
print.top_items	41
prune	42
set_factor_labels	46
sim16	47
suggest_k	49
summary.ackwards	53
tidy.ackwards	54
top_items	57

Index	60
--------------	-----------

Description

Extracts factor or component solutions at levels 1 through k. A factor (or component) is a summary variable that stands in for a group of items that move together. It then characterises the hierarchy by computing the correlations between the factor scores of different levels. A factor score is each person's estimated standing on a factor. Each edge is computed from the scoring weights and the correlation matrix stored as `x$r`, and it is exact for that matrix. For `engine = "esem"`, lavaan fits a different matrix in two settings. With `cor = "spearman"` and `missing = "pairwise"` or `"listwise"`, `x$r` is a Spearman matrix, but lavaan fits Pearson covariances of the raw data. With `cor = "pearson"` and `missing = "pairwise"` on data with missing values, `x$r` holds pairwise correlations. There, the ML, MLR, and ULSMV estimators fit the complete rows, and WLSMV fits the pairwise covariances. The correlations of those covariances differ slightly from `x$r`, because each item's variance uses all of its observed rows. 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,
  rotation = "varimax",
  ...
)
```

Arguments

<code>data</code>	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, <code>engine</code> must be <code>"pca"</code> or <code>"efa"</code> , because ESEM (exploratory structural equation modeling) needs raw data. The missing and
-------------------	--

	cor arguments are then ignored. See the <i>Correlation-matrix input</i> section below.
k_max	Maximum number of factors or components to extract. It is required, so use <code>suggest_k()</code> if uncertain. It sets the <i>depth</i> of the hierarchy: levels 1 through k_max are all extracted and retained. (The k_max of <code>suggest_k()</code> means something related but different. It is the largest number of factors that function <i>evaluates</i> when recommending a depth, not a depth itself. See that function's docs.)
engine	Extraction engine: "pca" (default), "efa", or "esem". "esem" uses <code>lavaan::efa()</code> with rotation-aware SEs and per-level fit indices. ESEM is exploratory structural equation modeling, and it is recommended for the clinical or HiTOP workflow (Kim & Eaton, 2015, Forbush et al., 2024). It requires lavaan >= 0.6-13.
cor	Correlation basis: "pearson" (default), "spearman", or "polychoric". For PCA and EFA, "polychoric" computes a polychoric correlation matrix via <code>psych::polychoric()</code> (requires psych). Polychoric correlations estimate the correlation between the continuous traits assumed to underlie ordered responses. For ESEM, it triggers WLSMV estimation via lavaan. A cli warning is emitted when ordinal-looking columns are detected and cor is not "polychoric". Ordinal items have a few ordered categories, such as a 1 to 5 rating.
fm	Factor extraction method passed to <code>psych::fa()</code> , and used only when engine = "efa". One of "minres" (the default, an ordinary least squares fit that converges reliably), "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" (maximum likelihood with standard errors that tolerate non-normal data). cor = "polychoric" with estimator = "ML"/"MLR" errors, because lavaan itself does not support ML or MLR on ordered indicators. "WLSMV"/"ULSMV" with a continuous cor is allowed, and gives a valid, if atypical, continuous WLS or ADF estimator. Ignored for PCA and EFA engines. The effective value (after auto-selection) is recorded in <code>x\$meta\$estimator</code> (NA for PCA and EFA).
missing	How to handle missing item responses. One of: "pairwise" (the default) uses all available observations pairwise. For PCA and EFA this feeds <code>stats::cor(use = "pairwise.complete.obs")</code>. For ESEM with WLSMV, and with ULSMV on cor = "polychoric", lavaan uses <code>available.cases</code>, which computes its sample statistics from all rows that contribute to each pair. That is valid under MCAR and uses the full N. For ESEM with ML, MLR, or ULSMV on a continuous cor, lavaan fits the complete rows, while edges are computed from a pairwise correlation matrix. The fields <code>\$meta\$missing</code> and <code>\$meta\$n_complete</code> record the setting and the number of complete rows. A warning is emitted when incomplete rows are detected. "listwise" uses only complete rows. It reduces the data to <code>stats::complete.cases()</code> before fitting, so the correlation matrix, the engine fit, and the edges all use the same rows. Under engine = "esem" with cor = "spearman", lavaan still fits a Pearson matrix (see the description above). <code>n_obs</code> in the result reflects the reduced N.

	• "fiml" is full information maximum likelihood. For engine = "esem" (with estimator = "ML"/"MLR" and cor = "pearson"), it passes missing = "fiml" to <code>lavaan::efa()</code> and derives edge correlations from lavaan's FIML-estimated saturated model. For engine = "pca"/"efa" (M38), the correlation matrix is estimated via <code>psych::corFiml()</code> and fed to the usual $W'RW$ algebra. That route is announced via a cli message. Both routes require cor = "pearson", because corFiml and lavaan's FIML each estimate a multivariate-normal matrix. FIML errors for WLSMV and ULSMV, and for a non-Pearson basis on any engine. FIML improves estimation under missingness but does not impute item responses. So score materialisation (keep_scores = TRUE) still produces NA rows for incomplete observations. See n_obs for the fit-index sample size on the PCA and EFA path.
n_obs	Number of observations. On the raw-data FIML path it may instead be a string selecting which N feeds the fit indices. FIML is full information maximum likelihood, which uses every observed value without dropping incomplete rows. • Correlation-matrix input: a positive integer. It is required for engine = "efa", because <code>psych::fa()</code> needs N for chi-square, RMSEA, and TLI. It is 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 <code>nrow(data)</code> and a numeric n_obs is ignored (with a warning). The exception is missing = "fiml" with engine = "pca"/"efa" (M38). <code>psych::corFiml()</code> estimates the correlation matrix from incomplete rows, so n_obs may be "total" or "complete". The default is "total", meaning every row that contributes to the FIML likelihood, which matches the FIML convention (Enders, 2010). "complete" is the complete-case N, a conservative lower bound. Point estimates do not depend on this choice. Those are the loadings (how strongly each item reflects a factor) and the edges. Only the fit indices of EFA (exploratory factor analysis) do. Those indices are <i>approximate</i> whatever N you pick, because of this two-step route: a FIML matrix fed into normal-theory EFA (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 <code>augment.ackwards()</code>). 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 <code>augment.ackwards()</code>).
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 steps that start at random: every lavaan rotation (ESEM) and psych's oblimin rotation for EFA. Both rotate from several random starts and keep the best one. Without a seed, two oblimin EFA fits of the same data can reach a different solution at a deep level. It is not used by PCA but is captured for reproducibility metadata. Default NULL.

pairs	Which level pairs to compute edges for. "adjacent" is the default and is the classic Goldberg choice of consecutive levels only. "all" is the Forbes extension, which takes every pair of levels, including skip-level correlations. The <code>prune()</code> function recomputes its own all-pairs edges on demand whatever this setting is, so pruning does not require <code>pairs = "all"</code> here.
cut_show	Edges with $ r \geq \text{cut_show}$ are flagged above_cut in <code>tidy()</code> output. Default 0.3.
correct	Continuity correction passed to <code>psych::polychoric()</code> on the PCA/EFA polychoric path (<code>engine = "pca"/"efa"</code> with <code>cor = "polychoric"</code>). Default 0.5 (psych's own default), which adds that value to zero cells before estimating thresholds. Set <code>correct = 0</code> if <code>psych::polychoric()</code> fails on your data, and its error suggests exactly this. That failure typically happens when an item has a near-empty response category, or when items with unequal category counts produce a sparse cross-cell. It is ignored on other paths: ESEM computes its own polychoric correlations inside lavaan, and the Pearson/Spearman bases do not use it.
rotation	The rotation used at every level with two or more factors. The default, "varimax", keeps the factors within a level uncorrelated. The other values are oblique rotations, which let them correlate. With <code>engine = "pca"</code> or <code>"efa"</code>, they are "oblimin" and "promax". With these engines "oblimin" needs the GPArotation package, and so does "promax" with <code>engine = "efa"</code>. With <code>engine = "esem"</code>, they are "oblimin" and "geomin" (lavaan's oblique geomin). With PCA, "promax" is <code>stats::promax()</code>. With EFA it is <code>psych::Promax()</code> through <code>psych::kaiser()</code>. The two can give slightly different loadings for the same solution. Under an oblique rotation, each edge <code>r</code> is a total correlation. Primary parents and signs still use <code>r</code>, and <code>tidy.ackwards()</code> reports the partialled beta beside it. Each level's <code>factor_cor</code> holds the engine's factor correlation. EFA and ESEM scores use the oblique form of the ten Berge weights, and PCA scores are the oblique components, so the scores correlate as the factors do. The variance values follow psych's convention for correlated factors. The fit announces what its edges mean, and <code>prune()</code> warns that its thresholds were set for varimax. Two kinds of rotation start at random: psych's oblimin for <code>engine = "efa"</code> and every lavaan rotation. Set seed to reproduce such a fit exactly (see seed). A rotation that fails is handled by how many starts it uses. PCA rotates once, so a level whose oblique rotation fails to converge, or that psych replaces with promax, ends the hierarchy at the level before it with a warning. So does a PCA level whose oblique fit raises an error. EFA (psych's <code>fa()</code> defaults to several starts) rotates from several random starts and keeps the best one, so an oblique rotation warning there can come from a discarded start. It is shown, and the level is kept. An EFA level whose final rotation step failed still ends the hierarchy, and so does any EFA rotation warning when the installed psych's <code>fa()</code> rotates from one start. ESEM counts the failed starts under every lavaan rotation, varimax included. At each rotated level, lavaan keeps the best of 30 random starts. This is lavaan's default, which <code>ackwards()</code> does not change. If some starts did not converge, a warning gives the count (for example, "3 of 30"), and the level is kept. If no

start converged, the kept rotation did not converge either, so the hierarchy ends at the level before it with a warning.

... 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"` is the original Goldberg (2006) method. PCA (principal component analysis) is the fastest engine and never fails to converge. Waller (2007) showed that the correlations between the component scores of different levels follow from the correlation matrix alone. (An oblique rotation can still fail, see `rotation`.)
- `rotation = "varimax"` keeps the within-level factors mutually uncorrelated (orthogonal). A rotation re-orientes the factors without changing how well they fit, and varimax pushes each item toward one factor. With uncorrelated factors, each between-level edge equals that ancestor's unique contribution to the descendant. That is what lets the edges be read as a lineage diagram. The closed-form $W'RW$ edge algebra is itself exact for the stored $x\$r$ under any fixed linear scoring, orthogonal or not. So the choice is interpretive, not a numerical necessity. It matches Goldberg (2006), Kim & Eaton (2015), and Forbush et al. (2024). An oblique rotation is available as a non-default option (see `rotation`). Its edges are total correlations, which also carry overlap through correlated factors at the same level.
- `cor = "pearson"` means 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. Ordinal items have a few ordered categories, such as a 1 to 5 rating. Polychoric correlations estimate the correlation between the continuous traits assumed to underlie those ordered responses.
- `align_signs = TRUE` because unaligned signs make the output unreadable. The anchor is `m1f1`, which 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` for memory and privacy. Scores are $O(n \times \Sigma_k)$ and often sensitive, and 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) lavaan derives costly sample statistics from the raw data. Those are the thresholds, the polychoric correlation matrix, and the asymptotic weight matrix. They 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, which gives identical solutions for 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, so 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", because psych needs N for chi-square, RMSEA, and TLI. It is optional for "pca", where it is stored as NA if omitted and is used only for the N -based sampling-adequacy checks and the result metadata. PCA computes no N -dependent fit statistics.
- **cor argument:** ignored, because the basis is already determined by the matrix you supply. A warning is emitted if you set `cor` explicitly.
- **missing argument:** ignored, because 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_`, and printed as "(user-supplied matrix)".

When to trust the result

`ackwards()` raises most diagnostics as it fits, while `tidy()` and `summary()` raise the warnings about a level's within-level score correlation (the item below says which raises which). They fall into three tiers by what they mean for whether you should trust and report the solution:

Fatal, so 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 because of 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 converge, and you should 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. The loadings and edges then rest on a rank-deficient matrix. (For EFA the residual-based fallback inflates TLI and RMSEA, and for ESEM CFI comes back NA.) Trim redundant items, or use `missing = "listwise"`. On the polychoric basis you can instead collapse sparse categories or set `correct = 0`.

Caution, so interpret carefully. The solution exists but may be unstable:

- A **Heywood case** at a level, meaning a communality > 1 or a negative uniqueness. Check that level's loadings make substantive sense, and 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. That is fine for a quick look or `suggest_k()` screening, but report the final model on `cor = "polychoric"`.
- A **nearly singular within-level score correlation** (smallest eigenvalue $< 1e-2$) means the factors at that level are close to collinear, so the partialled beta for edges from that level is unstable. The `tidy()` method still reports beta and raises a warning that names the level. Fitting with `ackwards()` does not raise it. The package chose this cutoff as a numerical guard. It is not a published rule. If the correlation cannot be inverted at all, beta and r^2 are NA and a different warning names the level. Both `tidy()` and `summary()` raise that one.

Informational, and usually fine. Proceed, just be aware of 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
- Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3), 187–200. doi:10.1007/BF02289233
- Grice, J. W. (2001). Computing and evaluating factor scores. *Psychological Methods*, 6(4), 430–450. doi:10.1037/1082989X.6.4.430
- Beauducel, A., Hilger, N., & Kuhl, T. (2024). The trade-off between factor score determinacy and the preservation of inter-factor correlations. *Educational and Psychological Measurement*, 84(2), 289–313. doi:10.1177/00131644231171137
- Williams, A. L., Conway, C. C., Olino, T. M., Revelle, W., Zinbarg, R. E., & HiTOP Utility Workgroup. (2025). Testing criterion validity in hierarchical models of psychopathology: Comparison of latent-variable and factor-score approaches. *Clinical Psychological Science*, 13(1), 128–145. doi:10.1177/21677026231225414

See Also

`print.ackwards()`, `tidy.ackwards()`, `glance.ackwards()`, `prune()`, which is the Forbes-extension flagging of redundancy (one factor adding nothing over another) and of artifacts, 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 to a data frame, for every level. A factor is a summary variable standing in for a group of items that move together, and a factor score is each person's estimated standing on one. 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). Supplying it with append = TRUE is an error, because all columns are already kept. Supplying it with data = NULL is also an error, because 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. That is 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 $S = Z W / \sqrt{\text{score\_var}}$, where Z is the item z-scores and W is the per-level weight matrix stored in the object. Dividing by $\sqrt{\text{score\_var}}$ standardizes by the real score standard deviations (Invariant 1: never assume unit variance). For PCA (principal component analysis) the method is "components". For EFA (exploratory factor analysis) and ESEM (exploratory structural equation modeling) it is "tenBerge". The scaling argument controls which means/SDs build Z. By default these are the **fit-time** moments stored in the object. Any data you score then lands on the same metric the model was estimated in: the training data, a subset of it, or new observations.

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. Setting `scaling = "sample"` instead re-standardizes by the supplied data's own moments. That is a deliberate choice when scoring a sample from a different population in its own metric. It is also the only option for objects fit from a correlation matrix, which carry no raw-data moments. For non-Pearson bases the usual caveat applies either way. Those bases are Spearman and polychoric correlations, which estimate the correlation between the continuous traits assumed to underlie ordered responses. 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 estimated by FIML, that is, full information maximum likelihood, which uses every observed value without dropping incomplete rows. 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. That is 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, the call `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 alignment holds. 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 `autoplot(x)` easy to call without a `library()` call.

Usage

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

Arguments

<code>object</code>	An object to visualise.
<code>...</code>	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. A factor is a summary variable standing in for a group of items that move together. Factors appear as labelled boxes arranged in levels. Level 1 sits at the top and level `k` at the 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,
  order = NULL,
  show_items = FALSE,
  n_items = 5,
  item_cut = 0.3,
  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_secondary = 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. There are four choices. The default "color" draws positive edges in color_pos and negative edges in color_neg. Option "linetype" draws positive as solid and negative as dashed. Option "both" uses colour <i>and</i> linetype, with negative drawn as a distinct double-dash so it reads clearly in greyscale. Option "none" does not encode sign and draws all edges color_edge and solid. Whichever channel is used gets a "Direction" legend. Colour is the default because it reads sign at a glance 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" (the default, where 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 < \text{cut_show}$ are not drawn. Defaults to the value used when the object was created (<code>x\$meta\$cut_show</code> , 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. Either a single value applied to every box (default 0.8), or a named numeric vector keyed by factor ID (e.g. <code>c(m5f1 = 1.6)</code>) giving a per-box width. Boxes not named fall back to the 0.8 default. Use a per-box width to fit a long manual node_labels string. Names matching no factor ID warn.
node_height	Height of factor boxes in layout units. Either a single value applied to every box (default 0.4), or a named numeric vector keyed by factor ID for a per-box height (unnamed boxes fall back to 0.4). This helps with multi-line manual labels. Names matching no factor ID warn.
min_sep	Minimum horizontal separation between nodes, passed to <code>ba_layout()</code> . Default 1.0.
order	Optional manual left-to-right ordering of the deepest (<code>k_max</code>) level, passed to <code>ba_layout()</code> . Supply a character vector of that level's factor IDs, or a named list keyed by the level number. That fixes the leaf order, and every upper factor re-centres above its primary children. Use it to untangle a figure by hand. NULL (default) uses the automatic ordering.
show_items	When TRUE, lists the salient observed items beneath each deepest-level (<code>k_max</code>) factor box, so a publication figure shows what each most-granular factor is made

	of. Items are the top <code>n_items</code> by loading (the item-factor correlation under varimax, or the pattern coefficient under an oblique rotation), taken in absolute value at or above <code>item_cut</code> . The extraction is the same as in <code>top_items()</code> . Variable labels are shown when the fit carried them, else the item IDs. Listed below the boxes in a vertical layout, and to their right in a horizontal one. Default FALSE.
<code>n_items</code>	Maximum number of items listed per deepest-level factor when <code>show_items = TRUE</code> . NULL lists every item meeting <code>item_cut</code> . Default 5.
<code>item_cut</code>	Absolute-loading threshold for <code>show_items</code> : only items with <code> loading >= item_cut</code> are listed. Default 0.3.
<code>show_skip</code>	Whether to draw skip-level (non-adjacent) edges. NULL (default) auto-detects: TRUE when the object was run with <code>pairs = "all"</code> , FALSE otherwise. Ignored when <code>drop_pruned = TRUE</code> .
<code>curvature</code>	Curvature of skip-level edge arcs. Passed to <code>ggplot2::geom_curve()</code> . Positive values curve right. Default 0.2. Ignored when <code>drop_pruned = TRUE</code> .
<code>color_pruned, colour_pruned</code>	Fill colour for nodes flagged as pruned or redundant (a factor that persists across levels without changing). Default "grey80". Only applied when the object carries pruning annotations (<code>x\$prune</code> is non-NULL) and <code>drop_pruned = FALSE</code> (pruned nodes are omitted entirely when <code>drop_pruned = TRUE</code>). <code>colour_pruned</code> is an accepted British alias.
<code>show_r</code>	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.
<code>r_digits</code>	Number of decimal places for edge labels when <code>show_r = TRUE</code> . Default 2L.
<code>r_label_size</code>	Font size for edge correlation labels when <code>show_r = TRUE</code> . Passed to <code>ggplot2::geom_label()</code> as <code>size</code> . Default 2.5.
<code>mono</code>	Monochrome convenience wrapper. When TRUE, equivalent to <code>sign_by = "linetype"</code> with black edges (it overrides <code>sign_by</code> and the colour arguments). <code>magnitude_by</code> still applies. Default FALSE.
<code>edge_linewidth</code>	NULL (default) uses <code>magnitude_by</code> to decide edge width. A numeric value forces every edge to that constant width (implying <code>magnitude_by = "none"</code>) and drops the <code> r </code> legend. Forbes figures use uniform thin lines (about 0.5 to 0.6).
<code>cut_strong</code>	Deprecated in M35 and ignored (with a warning). Edge magnitude is now shown by <code>magnitude_by</code> (linewidth) and sign by <code>sign_by</code> , because the old strong/weak linetype split double-encoded magnitude. Retained only so existing calls do not error.
<code>direction</code>	Layout orientation. "vertical" (default) stacks levels top-to-bottom, with level 1 at top. "horizontal" lays them out left-to-right, with level 1 at left, which suits wide slides or posters. Level-axis labels move to the bottom margin under "horizontal".
<code>show_level_labels</code>	Whether to draw level axis labels ("1 factor", "2 factors", ...) to the left of the diagram. Default TRUE. Sometimes the object carries pruning annotations

(`x$prune` non-NULL) and `drop_pruned = FALSE`. Then a level whose factors are *all* pruned has its axis label rendered in italic to denote its status. The italic matches the grey node fill. Partially-pruned levels keep a plain label.

<code>level_label_size</code>	Font size for level axis labels. Default 3.
<code>node_labels</code>	A named character vector mapping factor IDs (e.g. "m5f1") to custom display strings (e.g. "General"). Unspecified factors keep any label attached with <code>set_factor_labels()</code> , falling back to the default <code>m{k}f{j}</code> 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.
<code>primary_only</code>	When TRUE, only primary-parent edges (<code>is_primary == TRUE</code>) are drawn. Because skip-level edges are never primary, this also suppresses skip arcs. Ignored when <code>drop_pruned = TRUE</code> . Default FALSE.
<code>drop_pruned</code>	When TRUE, activates the Forbes (2023) pruned-view rendering path. Pruned nodes are removed from the diagram entirely. Each retained node is connected to its single strongest kept ancestor by a straight arrow, even across level gaps. This requires the object to carry pruning annotations (piped through <code>prune()</code>), and errors if not. Overrides <code>show_skip</code> , <code>curvature</code> , and <code>primary_only</code> . Default FALSE.
<code>compress_levels</code>	When TRUE under <code>drop_pruned = TRUE</code> , closes vertical gaps left by pruned levels so retained levels are evenly spaced. Level axis labels (<code>d</code>) still show the original level numbers. Ignored when <code>drop_pruned = FALSE</code> . Default FALSE.
<code>show_secondary</code>	When TRUE under <code>drop_pruned = TRUE</code> , also draws the between-level correlation edges the single-strongest-ancestor primary view hides: every kept cross-level factor pair with $ r \geq \text{cut_show}$ that is not already a primary edge. This includes cross-branch second parents <i>and</i> same-lineage skip arcs (a direct skip-level r is a distinct, non-transitive fact, not implied by the primary path). Secondary edges render in a channel deliberately distinct from the primary edges. They are dimmed (reduced opacity) and thinner, with plain line ends. They still inherit the sign encoding (<code>sign_by</code>), so the sign colour or linetype is never confused with the secondary channel. By construction each node's secondaries are weaker than its primary edge, so a secondary edge never appears while that node's primary is below <code>cut_show</code> . Ignored when <code>drop_pruned = FALSE</code> . Default FALSE.
<code>show_arrows</code>	When FALSE, edges are drawn with plain line ends instead of closed arrowheads (<code>arrow = NULL</code>). Applies to both straight and curved edge layers. Default TRUE. Forbes (2023) figures use plain line ends.
<code>legend</code>	When FALSE, suppresses all plot legends (<code>legend.position = "none"</code>). Useful when <code>color_pos == color_neg</code> (e.g. both "black") to remove an otherwise redundant Direction key. Default TRUE.
<code>...</code>	Ignored.
<code>x</code>	An <code>ackwards</code> object.

Details

When `what = "fit"`, renders a two-panel line plot of per-level fit indices. CFI and TLI go in the top panel, RMSEA and SRMR in the bottom panel. Horizontal reference lines mark the conventional Hu & Bentler (1999) thresholds. The anchor level ($k = 1$, saturated and always fits perfectly) is excluded. This needs the EFA (exploratory factor analysis) or ESEM (exploratory structural equation modeling) engine. It returns an informative empty plot for PCA (principal component analysis), 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**, because 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")

  # List the top items under each deepest-level factor
  autoplot(x, show_items = TRUE, n_items = 4)

  # 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"))

# Widen two boxes to fit long manual labels (per-box sizing)
autoplot(x,
  node_labels = c(m5f1 = "Conscientiousness"),
  node_width = c(m5f1 = 1.8)
)

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

# Manually order the deepest level to untangle the figure by hand
ids5 <- paste0("m5f", c(2, 1, 3, 5, 4))
autoplot(x, order = ids5)

# 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)

# Add the secondary between-level correlations the primary view hides
# (dimmed + thinner, drawn beneath the primary edges)
autoplot(xp, drop_pruned = TRUE, show_secondary = 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
)
}

```

Description

Renders a two-panel ggplot2 diagnostic for a `comparability()` object. The panels show score comparability (r) and loading congruence (Tucker's ϕ) for every factor at every level. A factor is a summary variable standing in for a group of items that move together. A loading is the correlation between an item and a factor, and congruence is a 0 to 1 index of how similar two loading patterns are. Grey points are individual splits, and black points are the per-factor medians. Dashed and dotted reference lines mark the conventional .90 and .95 benchmarks, which are visual guides, not tests.

Usage

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

Arguments

<code>object</code>	A comparability object.
<code>...</code>	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	<i>Plot a suggest_k diagnostic</i>
--------------------	------------------------------------

Description

Renders a ggplot2 diagnostic for a suggest_k object. The plot shows one panel for each criterion group that was requested. The groups are 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). PA is parallel analysis, which compares the eigenvalues of your data with those of random data of the same shape. When four panels are shown the layout is a 2x2 grid, and 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, and 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 a summary variable standing in for a group of items that move together. CD retains a factor while adding it significantly reduces RMSE (default $\alpha = 0.30$), and 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 is a layered (Sugiyama-style) barycenter algorithm in two stages:

Usage

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

Arguments

x	An ackwards object.
min_sep	Minimum horizontal separation between adjacent nodes at the same level. Default 1.0. Increase for wider diagrams.
order	Optional manual left-to-right ordering of the deepest (k_max) level, overriding Stage 1. Supply a character vector of the deepest level's factor IDs in the desired order. You may instead supply a named list with an entry for the deepest level, keyed by the level number as a string, e.g. <code>list("5" = c("m5f2", "m5f1", ...))</code> . Fixing the leaf order propagates upward: every upper factor stays at the mean-x of its primary children, so any arrangement of the primary forest is reachable from the leaf order. Entries for non-deepest levels are ignored with a warning (an upper factor's position is derived, not free). The vector must be a permutation of the deepest level's IDs. NULL (default) uses the automatic ordering.

Details

1. **Ordering** determines the left-to-right *order* of factors at each level. A factor is a summary variable standing in for a group of items that move together. Two candidate orderings are scored and the better one is kept. The first is a single top-down lrl-weighted barycenter sweep (the historical order). The second is a **primary-forest traversal**. Each factor has exactly one primary parent, so the primary edges form a forest. A depth-first, subtree-contiguous leaf order then lays every subtree out as an unbroken run. In deep hierarchies ($k \geq 10$) the traversal drives primary-tree crossings to zero, removing the "bent levels" a single pass leaves behind. The better ordering is the one with fewer primary crossings, and on a tie the one with fewer crossings overall, so shallow layouts are never made worse.

2. **X-assignment** assigns actual x coordinates bottom-up. The deepest level 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 the lrl-weighted mean of all children for factors with no primary children. Spreading resolves any remaining overlaps.

After both stages the layout is shifted so that the single level-1 node is always at x = 0. The result is fully deterministic.

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	<i>Big Five Inventory: 25-item IPIP example dataset</i>
-------	---

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 to 6, with some NAs reflecting genuine missing values in the original survey). The 25 items span the Big Five personality domains:

Columns	Domain
A1 to A5	Agreeableness
C1 to C5	Conscientiousness

E1 to E5 Extraversion
N1 to N5 Neuroticism
O1 to 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, so base row-subsetting (e.g. `na.omit(bfi25)`, `bfi25[rows, ]`) drops them. Base R does that for any vector that is not of class `labelled`. To keep the labels, fit on `bfi25` **directly**. Its NAs are handled by the missing argument of `ackwards()`.

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	<i>Bootstrap confidence intervals for between-level edges</i>
------------	---

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. This function quantifies the sampling uncertainty of that estimate. That matters most where a hard threshold consumes the estimate. Two such places are `prune()`'s `|r| >= redundancy_r` redundancy rule and the Forbes (2023) practice of interpreting the strongest all-pairs edge. A factor is a summary variable standing in for a group of items that move together, and a redundant factor is one that persists across levels without changing.

Usage

```
boot_edges(x, ...)

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

Arguments

x	An <code>ackwards</code> object fit with <code>engine = "pca"</code> or <code>"efa"</code> on a "pearson" or "spearman" basis. ESEM objects (exploratory structural equation modeling) are not supported, because refitting <code>n_boot</code> lavaan hierarchies needs its own performance treatment. Polychoric-basis objects are not supported either, because estimating a polychoric matrix in every resample is slow and unstable. Polychoric correlations estimate the correlation between the continuous traits assumed to underlie ordered responses. Objects fit from a correlation matrix are also unsupported, because resampling needs rows.
...	Reserved for future arguments.
data	The raw item data the model was fit on. Required, because the <code>ackwards</code> object deliberately does not store raw data (light core), so it must be re-supplied here. Columns are matched by name against the fit, and 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** before its edges are computed. That is, its factors are matched (greedy max-|r| with removal) and sign-oriented against the full-sample factors, on the full-sample correlation matrix. Without anchoring, factor label switching and sign flipping across replicates would corrupt the pooled edge distributions. This is the same matching machinery `comparability()` uses.

Each replicate is refit with the object's rotation. Only the edge `r` is bootstrapped, so the partialled beta and `r2` columns of `tidy.ackwards()` get no intervals. Under an oblique rotation the interval therefore describes the total correlation.

All resample indices are drawn upfront from `seed`, so results are reproducible. Replicate fits are dispatched through `future.apply` when it is installed, under the user's `future::plan()`, and otherwise run serially, as in the ESEM (exploratory structural equation modeling) engine of `ackwards()`. Most replicates are then fully determined by their resample indices, so serial and parallel runs agree exactly. The exception is an EFA object with `rotation = "oblimin"`: each refit also draws psych's random rotation starts. Its results are reproducible under `seed`. They can differ between a run with

future.apply installed and one without it, because the two draw the starts from different random streams.

Value

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

<code>edges</code>	Data frame with one row per edge, aligned with <code>tidy(x, what = "edges")</code> . The identifier columns are <code>from</code> , <code>to</code> , <code>level_from</code> , and <code>level_to</code> . The estimate columns are <code>r</code> (the full-sample point estimate), <code>se</code> (bootstrap standard error), and <code>lo</code> and <code>hi</code> (percentile interval endpoints). The column <code>n_boot_ok</code> counts the usable replicates for that edge.
<code>n_boot</code> , <code>conf</code> , <code>seed</code>	The request.

After calling `boot_edges()`, the table `tidy(x, what = "edges")` gains `se`, `lo`, and `hi` columns. Also, `print(x)` and `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. It is dropped from their distributions, because 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

Use `prune()` for the thresholded rules the intervals contextualise. See `comparability()` for split-half replicability of the factors themselves, where the sample is split into two random halves and each half is analysed separately. See `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. A factor is a summary variable standing in for a group of items that move together. Three columns cause most of the trouble. One is an item everyone answered the same way (no variance). Another is an item dominated by one response with a couple of stray answers. The third is an item with heavy missingness. These are especially costly on the polychoric basis, which estimates the correlation between the continuous traits assumed to underlie ordered responses. A near-empty response category can make `psych::polychoric()` fail outright (see the correct argument of `ackwards()`). 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` and `correct` deliberately rather than debugging a cryptic error. It only reports, and 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

<code>data</code>	A data frame or numeric matrix (items in columns).
<code>cor</code>	The correlation basis you plan to use: "polychoric" (default), "pearson", or "spearman". This only affects which problems are flagged, because 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). The last column, `flag`, is 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). Also `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. A factor is a summary variable standing in for a group of items that move together. In a split-half analysis the sample is split into two random halves and each half is analysed separately. The coefficients are Everett's (1983) factor comparability coefficients. They revive 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. This is also the direct instrument for the overextraction caution in [suggest_k\(\)](#), because 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 , because 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 or components to evaluate. Normally this is 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, because fitting $2 * n\_splits$ lavaan hierarchies needs its own performance treatment. For ESEM (exploratory structural equation modeling) workflows, run <code>comparability()</code> with <code>engine = "efa"</code> as a structural screen.
cor	Correlation basis: "pearson" (default) or "spearman". As with <code>suggest_k()</code> , "polychoric" is not supported, because estimating polychoric matrices in every half-sample is slow and unstable. Users analysing ordinal items (a few ordered categories, such as a 1 to 5 rating) should screen replicability on the Pearson basis. They should then fit the final model with <code>cor = "polychoric"</code> in <code>ackwards()</code> , which estimates the correlation between the continuous traits assumed to underlie the ordered responses.
fm	Factor extraction method passed to <code>psych::fa()</code> . It is only used when <code>engine = "efa"</code> . 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, so the coefficients are summarised over replicates. Each replicate fits $2 * k\_max$ solutions, so the default costs 20 hierarchy fits. PCA (principal component analysis) and EFA (exploratory factor analysis) 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 from 1 to `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()`. The comparability coefficient for a factor is the correlation between its two matched half-solution scores. It is 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 (ϕ) between the matched half-solution loading columns is reported alongside. A loading is the correlation between an item and a factor, and congruence is a 0 to 1 index of how similar two loading patterns are. So comparability asks whether the two halves' *scores* agree, and ϕ asks whether their *loading patterns* agree.

`comparability()` always uses the varimax rotation, the `ackwards()` default, for the full sample and for every half. It has no rotation argument, so its coefficients describe the varimax hierarchy even when you fit `ackwards()` with an oblique rotation.

Value

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

coefficients	Data frame with one row per split x level x factor. Its columns are <code>split</code> , <code>level</code> , <code>factor</code> (full-sample <code>m{k}f{j}</code> label), <code>r</code> (score comparability), and <code>phi</code> (Tucker's congruence of the matched loading columns). A value is NA when the level did not converge in one of the halves.
--------------	---

summary	Data frame with one row per level x factor. Its columns are 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. It can be lower than the k_max you asked for when the full-sample fit truncated, that is, when deep levels did not converge. 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. Two benchmarks are conventional. The first, **.90**, is the split-half replication threshold of Goldberg's lexical research program (Saucier et al., 2005). It follows from Everett's (1983) rationale, under which split-half factors share at least 81% of their variance. The second, **.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, because 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
```

factorability

Screen a dataset for factorability and sampling adequacy

Description

Two questions come before extracting a hierarchy. Is this correlation matrix even worth factoring, and is the sample large enough to trust the answer? A factor is a summary variable standing in for a group of items that move together. The function reports the standard diagnostics for both, in 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**, which tests whether the correlation matrix differs from the identity (via `psych::cortest.bartlett()`). It 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. It is a hard limit for EFA (exploratory factor analysis) and ESEM (exploratory structural equation modeling), though not for PCA (principal component analysis).

Read these as conventions, not verdicts. Three cutoffs appear here: Kaiser's KMO bands, the 5:1 and 10:1 N:p rules of thumb, and Bartlett significance at .05. Each 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 far more on two other things than on any fixed ratio (MacCallum et al. 1999). One is the communalities (the share of each item's variance the factors explain). The other is factor overdetermination (how many items load strongly on each factor). This function deliberately reports the numbers and their conventional bands rather than returning a pass/fail flag. `ackwards()` runs the same screen internally. It 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", which estimates the correlation between the continuous traits assumed to underlie ordered responses. 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. Its entries are `cor`, `n_obs`, `n_vars`, `np_ratio`, `kmo_overall`, `kmo_items` (a data frame of per-item MSA), `bartlett` (a list of `chisq`, `df`, and `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\(\)](#), which checks split-half replicability by splitting the sample into two random halves. Those are the other pre-analysis diagnostics. See also [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))
```

factor_labels	<i>Read the factor labels stored on an ackwards object</i>
---------------	--

Description

A factor is a summary variable standing in for a group of items that move together. This function returns the labels that `set_factor_labels()` stored on the 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)
```

forbes2023	<i>Assessing Mental Health symptom correlation matrix (Forbes 2023 applied example)</i>
------------	---

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. A factor (or component) is a summary variable standing in for a group of variables that move together. This is 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 to 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. That is 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 fit statistics scaled to the original sample for EFA (exploratory factor analysis) or ESEM (exploratory structural equation modeling).

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., Callear, 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 (exploratory factor analysis) and ESEM (exploratory structural equation modeling) 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. For example, CFI and SRMR are NA for EFA, and all five are NA for PCA (principal component analysis). For ESEM, BIC is NA under estimator = "WLSMV" or "ULSMV", because these limited-information estimators have no proper log-likelihood, and it is populated under "ML" or "MLR". Under a scaled-test estimator ("WLSMV", "ULSMV", or "MLR") the CFI, TLI, and 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	<i>Generate a node-label scaffold for autoplot</i>
----------------	--

Description

Returns a named character vector covering all factor IDs in the object, ready to pass to `autoplot(x, node_labels = ...)`. A factor is a summary variable standing in for a group of items that move together. Printing the result shows an editable `c(...)` literal. 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

<code>x</code>	An <code>ackwards</code> object.
<code>style</code>	One of "id" (default), "forbes", or "blank". See Style options above.
<code>...</code>	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*) makes every value equal the factor ID (" $m1f1$ ", " $m2f1$ ", ...). This is a round-trip no-op, because passing the result to `node_labels` without editing reproduces the default labels exactly. It is a good starting point for adding substantive labels.
- "forbes" gives values that follow the Forbes (2023) convention, a level-letter plus a 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). This requires `k_max <= 26`, because LETTERS has 26 entries. An error is raised for deeper objects.
- "blank" makes all values 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	<i>Score new observations with a fitted ackwards model</i>
------------------	--

Description

Applies a fitted bass-ackwards model to new data, for example the held-out test split of a cross-validation design. A factor is a summary variable standing in for a group of items that move together, and a factor score is each person's estimated standing on one. This function gives factor scores at every level **without retraining**. It 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

<code>object</code>	An ackwards object.
<code>newdata</code>	A data frame or numeric matrix with the same variables (columns) used to fit object. Required. To retrieve scores stored at fit time, use <code>augment(object)</code> instead.
<code>scaling</code>	Which item means/SDs standardize newdata: "fit" (default, the training moments) or "sample" (newdata's own moments). See augment.ackwards() .
<code>...</code>	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. The new scores then 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. They are matched by column name, and extra columns are 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	<i>Print an ackwards object</i>
----------------	---------------------------------

Description

Displays a compact summary of the bass-ackwards result using cli formatting. It prints no matrix dumps. Use [tidy.ackwards\(\)](#) to read values in code.

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	<i>Print an item quality check</i>
-------------------	------------------------------------

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	<i>Print a comparability object</i>
---------------------	-------------------------------------

Description

Print a comparability object

Usage

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

Arguments

x	A comparability object.
...	Ignored.

Value

x invisibly.

print.factorability	<i>Print a factorability screen</i>
---------------------	-------------------------------------

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	<i>Print a suggest_k object</i>
-----------------	---------------------------------

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` and `prune_reason` flags in `x$prune$nodes`. A factor is a summary variable standing in for a group of items that move together. A redundant factor is one that persists across levels without changing. Pruning is flag-only and never removes, so 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
```

The function `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,
  near_margin = 0.1,
  ...
)
```

Arguments

<code>x</code>	An <code>ackwards</code> object.
<code>...</code>	Reserved for future methods/arguments.
<code>rules</code>	Character vector controlling which auto-rules run. Default <code>"none"</code> , which runs no auto rule. Combine it with <code>manual</code> for pure manual pruning, or call <code>prune(x)</code> with no arguments to clear any existing pruning. Options:

- "redundant": identify chains of factors connected by factor-score correlations $|r| \geq \text{redundancy\_r}$ (and optionally $\phi > \text{redundancy\_phi}$), using `redundancy_criterion` (default "direct", which follows Forbes's reference code). A factor score is each person's estimated standing on a factor. Applies Forbes's (2023) retention rule: keep the bottom node when the chain reaches level `k_max` (most specific), and keep the top node otherwise. Flagged nodes get `pruned = TRUE` and `prune_reason = "redundant"` in `x$prune$nodes`.
 - "artifact" (or the alias "artefact", normalized to "artifact"): compute Tucker's congruence coefficient (ϕ) for all cross-level factor pairs and store it in `x$prune$phi`. Congruence is a 0 to 1 index of how similar two loading patterns are. A loading is the correlation between an item and a factor under varimax, and the pattern coefficient under an oblique rotation. This mode also stores structural signals (`few_items`, `orphan`, and `split_merge`) in `x$prune$structural`, and the **near-redundant band** (`near_margin`, see Details) in `x$prune$near_redundant`. No factors are auto-flagged, because 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`, and 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:
- `x$engine == "pca"`: no ϕ filter. PCA is principal component analysis, and a component is a weighted sum of the items. Component scores are *determinate*, that is, exact linear functions of the data, with no factor-score indeterminacy. So the score correlation $|r|$ is the correlation between the components themselves under the correlation matrix the fit uses, and ϕ adds nothing that $|r|$ does not already capture.
 - `x$engine` is "efa" or "esem": automatically set to 0.95 (Lorenzo-Seva & ten Berge, 2006). EFA is exploratory factor analysis and ESEM is exploratory structural equation modeling. Factor-score indeterminacy off-PCA means $|r|$ alone is liberal, so the conjunctive ϕ criterion is the conservative default. A cli message announces the resolved value. Pass NA to disable ϕ 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. At each level it takes the factor with the largest direct $|r|$. It continues while $|r| \geq \text{redundancy\_r}$, and while ϕ is above `redundancy_phi` when `redundancy_phi` is set (by you or automatically). The chase stops at the first level that fails. A level further up that passes again does not restart it. This contiguous stop

follows Forbes's (2023) reference code, ChaseCorrPaths, and the chase reproduces her AMH applied example exactly. The direct correlation is the honest way to operationalize "the same construct", because two factor scores share $\geq \text{redundancy\_r}^2$ variance *directly*. To bridge such a gap, judge the case yourself, including which factor of the bridged chain to drop, and pass that factor to manual. A factor that no rule flagged then gets `prune_reason = "manual"`, and a factor already flagged as redundant keeps that reason. `manual` adds no row to `x$prune$chains`.

- "adjacent": trace **adjacent primary-parent** links only, so each consecutive level must meet $|r| \geq \text{redundancy\_r}$. 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 or transitive hierarchies the two agree.

<code>min_items</code>	Minimum number of items for which a factor must be the primary loader (highest <code> loading </code>). Factors with fewer than <code>min_items</code> primary items are flagged <code>few_items = TRUE</code> in <code>x\$prune\$structural</code> . Only used when rules includes "artifact". Default 3L, because 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).
<code>orphan_r</code>	Threshold for the orphan structural signal. A factor whose maximum adjacent-level $ r $ (to the immediately shallower and deeper levels) falls below <code>orphan_r</code> is flagged <code>orphan = TRUE</code> in <code>x\$prune\$structural</code> . Such a factor 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, because a factor that shares less than a quarter of its variance with every neighbour is a structural outlier worth inspecting.
<code>near_margin</code>	Scalar in (0, 1]. Width of the near-redundant band reported by "artifact" mode (see Details). A cross-level pair is flagged near-redundant when its direct $ r $ sits in $[\text{redundancy\_r} - \text{near\_margin}, \text{redundancy\_r})$ or its Tucker phi sits in $[\text{redundancy\_phi} - \text{near\_margin}, \text{redundancy\_phi})$. That is, it sits within <code>near_margin</code> <i>below</i> a redundancy threshold, while the pair is not itself fully redundant. Only used when rules includes "artifact". Default 0.1.

Details

Oblique objects. When `ackwards()` was fit with an oblique rotation, each edge r is a total correlation, which includes overlap through correlated factors at the same level. Every rule still runs unchanged: the edge criteria on r , and the congruence (ϕ) checks and item-assignment signals on the loadings, which are then pattern coefficients. A warning names the rules that run and says that they assume orthogonal levels. Their default thresholds, such as `redundancy_r = 0.9`, were calibrated on varimax edges.

The "direct" criterion is a star anchored on the leaf, not a walk. Take a three-level chain candidate with deepest leaf `m3f1` and shallower factors `m2f1` (level 2) and `m1f1` (level 1). Under the default "direct" criterion the chain `m1f1 -> m2f1 -> m3f1` forms when **both** direct-to-leaf correlations $|r(\text{m1f1}, \text{m3f1})|$ and $|r(\text{m2f1}, \text{m3f1})|$ meet `redundancy_r`. Every member is judged by its own direct correlation to the *same* deepest factor, a star centred on the leaf. It does **not** require

the adjacent hop $|r(m1f1, m2f1)|$ to meet the threshold (that is the "adjacent" criterion). It also does **not** screen every ancestor pair against every other, so there is no all-pairs test. An ancestor can join on a strong direct link to the leaf even where its adjacent hop to the next chain member is weak. That is exactly why `r_to_prev` (below) can sit under `redundancy_r`.

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`, because 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`.

The near-redundant band ("artifact" mode). `prune("redundant")` drops *full* redundancy, meaning pairs at or above the thresholds. Forbes (2023) uses the artifact flags mainly for the messier band *just below* them. Say a pair correlates at $|r| = 0.89$ and shares a loading pattern at $\phi = 0.93$. It is not quite redundant, but it may be the same factor re-rotated (re-oriented without changing fit), so it is worth a second look. Artifact mode surfaces this as `x$prune$near_redundant`. That data frame holds every cross-level pair that is **not** itself fully redundant. Such a pair has its direct (skip-level) $|r|$ **or** its Tucker ϕ within `near_margin` *below* the corresponding threshold (`redundancy_r` or `redundancy_phi`). Its columns are `from`, `to`, `level_from`, `level_to`, `r` (direct score correlation), `phi` (loading congruence), and the logical band flags `near_r` and `near_phi`. Under the PCA engine `redundancy_phi` is NULL (component scores are determinate), so only the $|r|$ band applies. Under EFA and ESEM `redundancy_phi` auto-resolves to 0.95 (announced via cli) and the ϕ band is active too. Like ϕ and the structural signals, the band is **report-only**. It never drops or mutates the kept node set.

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	<i>Attach persistent factor labels to an ackwards object</i>
-------------------	--

Description

A factor is a summary variable standing in for a group of items that move together. 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 or 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, because 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, and 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. A factor is a summary variable that stands in for a group of items that move together. The dataset showcases the default `cor = "pearson"` extraction path without the ordinal-detection warning that `bf125`'s Likert items trigger. Ordinal items have a few ordered categories, such as a 1 to 5 rating.

Usage

```
sim16
```

Format

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

Details

Population model. $\Sigma = \Lambda \Lambda^T \Phi \Lambda + \Psi$, an oblique common-factor model with 4 true group factors, sampled via a Cholesky factorization (base R, with no MASS dependency):

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

All items load 0.75 on their true factor, with no cross-loadings. A loading is the correlation between an item and a factor. Factor correlations are 0.45 within a metatrait (f1-f2, f3-f4) and 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 reach a consensus of $k = 4$, from five criteria, because VSS reports at complexities 1 and 2.

Idealized by design. The planted signal is strong and clean, so all six `suggest_k()` recommendations converge on $k = 4$. That is 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 $k = 4$ to 6. The two datasets are complementary teaching foils. Use `sim16` for "watch the method recover a structure we planted," and `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 (exploratory factor analysis) 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`. The true factors, the ones that do not split, persist almost unchanged from $k=3$ onward. So their parent-child score correlations approach 1 and are flagged by `prune(x, "redundant")` ($|r| \geq .9$ and, under the EFA auto-default, Tucker's $\phi > .95$). Two factors are redundant when one adds nothing over the other. This is a textbook overextraction artifact, included so the Forbes and 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. A factor is a summary variable standing in for a group of items that move together. 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, <i>n_obs</i> is required (PA and VSS need <i>N</i>) and the <i>cor</i> argument is ignored. The Comparison Data (CD) criterion is then skipped, because CD requires raw item distributions for resampling.
k_max	Maximum number of factors or components to <i>evaluate</i> when recommending a depth. It is not a depth itself. Defaults to <code>min(ncol(data) - 1, 8)</code> . Increase if you expect a deeper hierarchy. (One caution: this is a distinct meaning from <code>ackwards()</code> 's <i>k_max</i> , which <i>is</i> the extraction depth. The two share a name because they are the same dial in the <code>suggest_k()</code> -> <code>ackwards()</code> workflow, just applied at different stages.)
criteria	Character vector of criteria to compute. Any subset of <code>c("pa_pc", "pa_fa", "map", "vss", "cd")</code> , and the default is all five. The criteria "pa_pc" and "pa_fa" share one parallel-analysis call, so both or neither are fast, and "map" and "vss" share one VSS call. Criteria not requested are skipped entirely (no computation) and their <i>k_*</i> fields in the result are NA. The choice "vss" selects both VSS-1 and VSS-2 as a unit.
cor	Correlation basis: "pearson" (default) or "spearman". Should match the <i>cor</i> 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 <code>nrow(data)</code>).
n_iter	Number of Monte Carlo iterations for parallel analysis. Default 20. Reduce to 5 for fast or exploratory runs, and increase to 100+ for publication.
seed	Integer seed passed to <code>set.seed()</code> before the Comparison Data (CD) step. NULL (default) uses the current RNG state. Note: the parallel-analysis step uses <code>psych::fa.parallel()</code> , which does not respond reliably to <code>set.seed()</code> . 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. Parallel analysis compares the eigenvalues of your data with those of random data of the same shape. It suggests retaining components (weighted sums of the items) whose eigenvalues exceed the 95th percentile of chance. It tends to overextract, so treat it 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 (exploratory factor analysis) and ESEM (exploratory structural equation modeling) 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, where a loading is the correlation between an item and a factor. 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. It 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, or call `autoplot()` on it for a diagnostic scree/criteria plot. The list contains:

k_parallel_pc	Recommended k from PC-based parallel analysis. It is NA_integer_ when "pa_pc" is not in <code>criteria</code> .
k_parallel_fa	Recommended k from FA-based parallel analysis. It is NA_integer_ if no FA factor exceeded the random threshold, or when "pa_fa" is not in <code>criteria</code> .

k_map	Recommended k from MAP. It is NA_integer_ when "map" is not in criteria.
k_vss1	Recommended k from VSS complexity-1. It is NA_integer_ when "vss" is not in criteria.
k_vss2	Recommended k from VSS complexity-2. It is NA_integer_ when "vss" is not in criteria.
k_cd	Recommended k from Comparison Data. It is NA_integer_ when "cd" is not in criteria, EFAtools is not installed, or CD fails.
cd_available	Logical. It is TRUE when "cd" was requested, EFAtools was found, and CD ran successfully.
criteria	Data frame with one row per k. Its columns are k, ev_obs (observed PC eigenvalue), ev_obs_fa (observed FA eigenvalue), pa_pc_quant and pa_fa_quant (95th-percentile simulated eigenvalue for each basis), pa_pc_suggested and pa_fa_suggested (logical retention), map, vss1, and 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. 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. Polychoric correlations estimate the correlation between the continuous traits assumed to underlie ordered responses. Ordinal items have a few ordered categories, such as a 1 to 5 rating. So when the raw data look ordinal (at most 7 distinct integer values per column) this function emits a one-per-session warning. That is the same `detect_ordinal()` signal `ackwards()` and `comparability()` use (Invariant 6: announce consequential defaults loudly). The advice points at the *final* `ackwards()` fit (`cor = "polychoric"`), **not** at `suggest_k()` itself, because the screening basis is intentional. The plausible-k *range* changes little whether you use Pearson or polychoric correlations. 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
- Tong, L., Qu, W., & Zhang, Z. (2025). Comparison of the K1 rule, parallel analysis, and the Bass-Ackward method on identifying the number of factors in factor analysis. *Fudan Journal of the Humanities and Social Sciences*, 18(1), 17–44. doi:10.1007/s40647024004232
- 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()`. See also `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	<i>Summarise an ackwards object</i>
------------------	-------------------------------------

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. It says more than `print.ackwards()` and is designed for inspection and reporting.

Usage

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

Arguments

object	An ackwards object.
...	Ignored.

Details

A "Within-level factor correlations" block lists, for each level, the correlation between every pair of factors at that level (the same values as `tidy(x, what = "factor_cor")`). The block appears only when some pair is correlated, that is when at least one within-level correlation exceeds $1e-8$ in size. Under the default varimax rotation the factors within a level are uncorrelated, so the block is absent. Under an oblique rotation the block lists the engine's factor correlations, and a note under the lineage says that the edges are total correlations.

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. A factor is a summary variable standing in for a group of items that move together, and a loading measures how strongly an item reflects a factor. Under the default varimax rotation a loading is the correlation between the item and the factor. Under an oblique rotation it is a pattern coefficient, the item's regression weight on the factor.

Usage

```
## S3 method for class 'ackwards'
tidy(
  x,
  what = c("edges", "loadings", "variance", "fit", "nodes", "scores", "factor_cor"),
  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, with columns from, to, level_from, level_to, r, beta, is_primary, above_cut. The column r is the correlation between the two factors' scores implied by the fit's correlation matrix, a total correlation. Under an oblique rotation it includes overlap through correlated factors at the from level, and is_primary still marks each factor's largest |r|. The column beta is the partialled coefficient. It is the standardized regression weight of the to factor on all factors of the from level together. It removes the part of r that the other factors at the from level share. Under the default varimax rotation the factors within a level are uncorrelated, so beta equals r. The two come apart only when the factors within a level are correlated. The column beta uses the within-level score correlation of the from level. That correlation comes

from the stored score weights and the fit's correlation matrix, the same matrix that `r` comes from. When that correlation cannot be inverted, `beta` is NA for that level's edges and a warning names the level. When that correlation is nearly singular (smallest eigenvalue below $1e-2$), the factors at that level are close to collinear and `beta` is unstable. The `tidy()` method still reports `beta` and raises a warning that names the level. Fitting with `ackwards()` does not raise it. The package chose this cutoff as a numerical guard. It is not a published rule. With the default ten Berge or component scores, the within-level score correlation equals the level's factor correlation (what = "factor_cor"). With regression scores, the fallback when ten Berge weights fail, the two can differ. 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, with columns `level`, `factor`, `item`, `loading`, `se`, `ci_lower`, `ci_upper`. The columns `se`, `ci_lower`, and `ci_upper` are populated only for `engine = "esem"`. That engine produces loading SEs that account for the rotation, which is the step that turns the raw solution into a more interpretable one. They are NA for PCA (principal component analysis) and EFA (exploratory factor analysis). The confidence level is controlled by `conf_level`.
- "variance": one row per factor x level, with columns `level`, `factor`, `proportion`, `cumulative`, `r2`. The first two are proportions of total item variance on a 0-1 scale (multiply by 100 for a percentage). Under an oblique rotation `proportion` follows psych's convention for correlated factors ($\text{diag}(\Phi L'L) / p$). The shares still sum to the level's total common variance, so each level's last cumulative value equals its varimax value. The running values before it can differ. A factor's share is then not its unique contribution. The column `r2` is the share of the factor's score variance that all factors of the level just above account for together. It is also a proportion on a 0-1 scale, but of that factor's own score variance, not of total item variance, so it is not comparable with `proportion` or `cumulative`. It is NA at level 1, which has no level above. Under the default varimax rotation `r2` equals the sum of the squared `r` values of that factor's edges from the level above. It is NA, with a warning, when the level above's within-level score correlation cannot be inverted. A nearly singular one raises no warning here. That warning comes from what = "edges", because it concerns `beta`.
- "factor_cor": one row per pair of factors within a level, with columns `level`, `factor_a`, `factor_b`, `cor`. The column `cor` is the correlation between the two factors as the engine reports it, in the stored column order and sign. Level 1 has one factor and contributes no row. Under the default varimax rotation every `cor` is 0. Under an oblique rotation it is psych's Φ or lavaan's latent correlation.
- "fit": one row per fit statistic x level, with columns `level`, `statistic`, `value`. For PCA objects the statistics are eigenvalues. For EFA objects they are `chi`, `dof`, `p_value`, `RMSEA`, `TLI`, and `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 (exploratory structural equation modeling) they are chi, dof, p_value, CFI, TLI, RMSEA, SRMR, BIC. Three estimators run a scaled test: "WLSMV" and "ULSMV" for ordinal items (a few ordered categories, such as a 1 to 5 rating), and "MLR" for continuous ones. Under any of them the whole row reports lavaan's mean-and-variance-adjusted ("scaled") variant, so every quantity shares one scaling. That covers chi, dof, and p_value **and** CFI, TLI, and RMSEA. This matters most for WLSMV and ULSMV. The naive chi-square has no valid reference distribution (lavaan's own summary() labels its p-value "Unknown"), and the naive CFI and TLI are badly optimistic for ordinal data (Xia & Yang, 2019). The estimator "ML" has no scaled variant, so it reports the naive values (the correct ones for ML). The statistic SRMR has no scaled variant and is reported as-is. The statistic BIC is NA under WLSMV and ULSMV, because a limited-information estimator has no proper log-likelihood, and it is populated under ML and MLR. Use format = "wide" for one row per **non-anchor** level and one column per statistic. Non-anchor means $k \geq 2$, because the saturated 1-factor anchor is dropped, matching summary() and autoplot(what = "fit"). Conventional fit cut-offs (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). The format argument is oriented to the EFA and 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, with columns 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, where a factor score is each person's estimated standing on a factor (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). That is 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 $\text{loading} \pm \text{qnorm}((1 + \text{conf\_level}) / 2) * \text{se}$ 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"`, `from_label/to_label` for `what = "edges"`, and `factor_a_label/factor_b_label` for `what = "factor_cor"`. 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 = "factor_cor") # all 0 under varimax
tidy(x, what = "fit")
tidy(x, what = "fit", format = "wide")
```

top_items

Display the salient items for each factor

Description

A factor is a summary variable standing in for a group of items that move together. A loading measures how strongly an item reflects a factor. Under the default varimax rotation it is the correlation between the item and the factor. Under an oblique rotation it is a pattern coefficient, the item's regression weight on the factor. 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 <code>ackwards</code> object.
level	Integer vector selecting which level(s) to include. <code>NULL</code> (default) returns all levels.
cut	Absolute-loading threshold. Items with <code> loading >= cut</code> are shown. Default 0.3, matching the <code>cut_show</code> plotting default.
n	Maximum number of items to show per factor. <code>NULL</code> (default) shows all items meeting the cut. When set, the top-n items by <code> loading </code> are kept after applying the cut.
sort	Logical. When <code>TRUE</code> (default), items within each group are ordered by descending <code> loading </code> . Set to <code>FALSE</code> to keep the original order (useful when items have a meaningful sequence).
by	One of "factor" (default) or "item". Option "factor" groups the listing by factor, giving the salient items <i>of</i> each factor, which answers "what is this factor about?". Option "item" inverts the grouping, so for each item it lists the factors it loads on. That makes cross-loadings legible, and answers "where does this item go?". Both <code>n</code> and <code>sort</code> apply within whichever unit <code>by</code> selects.
show_labels	Logical. When <code>TRUE</code> (default) and the data carried a variable-label attribute at fit time (see ackwards()), items are shown as <code>id: label</code> . Items without a label fall back to the bare <code>id</code> . Set to <code>FALSE</code> to always show the bare <code>m{k}f{j}</code> -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. That is expected and informative.

If [factor labels](#) have been attached, the factor dimension is shown as `label (id)` wherever it appears. That covers 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

- * **datasets**
 - bfi25, 23
 - forbes2023, 33
 - sim16, 47
- ackwards, 3
- ackwards(), 12, 24, 25, 27–32, 38, 42, 44, 45, 49–52, 58
- augment(), 46
- augment.ackwards, 10
- augment.ackwards(), 5, 7, 37, 38, 56
- autoplot, 13
- autoplot(), 29, 46, 50
- autoplot.ackwards, 13
- autoplot.ackwards(), 13, 23, 36, 37, 45, 47, 58
- autoplot.comparability, 19
- autoplot.suggest_k, 21
- ba_layout, 22
- ba_layout(), 15, 18, 36, 37
- bfi25, 23
- boot_edges, 24
- boot_edges(), 46, 55
- check_items, 27
- check_items(), 8, 9, 32, 39, 52
- comparability, 28
- comparability(), 20, 25–27, 32, 52
- cor(), 12
- factor labels, 57, 58
- factor_labels, 33
- factor_labels(), 46, 47
- factorability, 31
- factorability(), 27, 30, 40, 52
- forbes2023, 33
- future::plan(), 25
- ggplot2::geom_curve(), 16
- ggplot2::ggsave(), 18
- glance.ackwards, 35
- glance.ackwards(), 7, 10, 39, 53, 57
- label_template, 36
- label_template(), 47, 58
- lavaan::efa(), 4, 5
- lm(), 12
- plot.ackwards (autoplot.ackwards), 13
- plot.ackwards(), 18
- predict(), 46
- predict.ackwards, 37
- predict.ackwards(), 11
- print(), 46
- print.ackwards, 38
- print.ackwards(), 7, 10, 35, 53, 57
- print.ackwards_labels (label_template), 36
- print.check_items, 39
- print.comparability, 39
- print.factorability, 40
- print.suggest_k, 40
- print.summary_ackwards, 41
- print.summary_ackwards(), 53
- print.top_items, 41
- prune, 42
- prune(), 6, 10, 17, 24, 26, 30, 46
- psych::corFiml(), 5
- psych::cortest.bartlett(), 31
- psych::fa(), 4, 5, 29, 55
- psych::KMO(), 31
- psych::polychoric(), 6, 27
- set.seed(), 50
- set_factor_labels, 46
- set_factor_labels(), 17, 33, 57, 58
- sim16, 47
- suggest_k, 49
- suggest_k(), 4, 21, 27–30, 32
- summary(), 46

summary.ackwards, [53](#)

tidy(), [46](#)

tidy.ackwards, [54](#)

tidy.ackwards(), [6](#), [7](#), [10](#), [12](#), [23](#), [25](#), [26](#), [35](#),
[38](#), [39](#), [45](#), [53](#), [58](#)

top_items, [57](#)

top_items(), [16](#), [37](#), [41](#), [46](#)