Package: bifactory (via r-universe)

November 1, 2026

Title (Bifactor) ESEM with Continuous (MLR) or Ordinal (WLSMV) Data

Version 0.6.2

Description Fits bifactor exploratory structural equation models (B-ESEM), together with standard exploratory structural equation modeling (ESEM) and confirmatory factor analysis (CFA), for continuous and ordinal data. Continuous models use 'lavaan' native efa() blocks with robust maximum likelihood (MLR) estimation. Ordinal ESEM defaults to the 'lavaan' weighted least squares mean- and variance-adjusted (WLSMV) estimator; ordinal B-ESEM uses a custom diagonally weighted least squares (DWLS) path with polychoric correlations from 'lavaan', rotation-delta standard errors via 'numDeriv', and a mean- and variance-adjusted chi-square. Target, geomin, and oblimin rotations use 'GPArotation'; the bifactor ESEM approach follows Morin, Arens and Marsh (2016) <doi:10.1080/10705511.2014.961800>. Additional features include multi-group measurement invariance (configural through strict, with partial invariance), ESEM-within-CFA conversion, McDonald's omega reliability suite, and the Mehrvarz and Rouder (2026) <doi:10.31234/osf.io/95enc_v3> alignment ratio check for independent cluster model confirmatory factor analysis (ICM-CFA) misspecification. An optional 'MplusAutomation' interface allows side-by-side comparison with 'Mplus' output.

License AGPL-3

URL https://github.com/leondebeer/bifactory

BugReports https://github.com/leondebeer/bifactory/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports lavaan (>= 0.7-2), GPArotation, psych, MASS, numDeriv, stats, utils, graphics, methods, parallel, withr

Suggests MplusAutomation, openxlsx2, pkgload, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Leon T. De Beer [aut, cre]

Maintainer Leon T. De Beer <leondb@gmail.com>

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

Date/Publication 2026-11-01 12:12:57 UTC

RemoteUrl https://github.com/leondebeer/bifactory

RemoteRef HEAD

RemoteSha 46b62267d3629c9bece69de896d9cca3356c8d02

Contents

align_loadings . . . 3
alignment_check . . . 5
besem . . . 8
besem_ordered . . . 12
bifactory_template . . . 14
chisq_decomp . . . 15
coef.esem_fit . . . 16
compare_ewc . . . 17
compare_loadings . . . 18
compute_indices . . . 19
compute_omega . . . 20
esem . . . 21
esem_compare . . . 25
esem_invariance . . . 26
esem_ordered . . . 30
ewc_syntax . . . 34
extract_mplus_loadings . . . 35
factor_correlations . . . 36
factor_scores . . . 36
find_ewc_referents . . . 38
fit_ewc . . . 39
fit_indices . . . 40
fitMeasures.esem_fit . . . 41
generate_mplus_besem_syntax . . . 42
generate_mplus_syntax . . . 43
get_syntax . . . 44
lavaan_fit . . . 45
make_bifactor_target . . . 45
make_target . . . 46
modindices.esem_fit . . . 48
parameters . . . 48
parse_mplus_polychoric . . . 49

partial_invariance 50
plot.alignment_check 52
print.alignment_check 52
print.besem_fit 53
print.bifactory_parameters 53
print.esem_comparison 54
print.esem_comparison_pipeline 54
print.esem_fit 55
print.esem_invariance 55
print.esem_partial_invariance 56
print.esem_spec 56
print.esem_target 57
print.ewc_comparison 57
print.ewc_fit 58
print.reliability_indices 58
refine_rotation 59
run_comparison 60
run_mplus_besem_invariance 62
save_results 63
set_target 64
specify_model 65
std_loadings 67
summary.esem_fit 67
summary.ewc_fit 68

Index **69**

`align_loadings` *Align ESEM/B-ESEM standardized loadings to a reference*

Description

Rotates a fitted solution's standardized loadings by an orthogonal Q so they match a reference matrix (Procrustes) or a deterministic canonical orientation. Useful when comparing bifactor solutions across software, or when you want a reproducible orientation across reruns.

Usage

```
align_loadings(
  x,
  target = "canonical",
  level = "configural",
  se_method = c("approx", "none")
)
```

Arguments

x	An esem_fit, besem_fit, esem_invariance, or raw lavaan S4 fit.
target	One of: • a numeric matrix (single-group) or list of matrices (multi-group) of reference standardized loadings, with rownames = items, colnames = factors; • another fit object of compatible structure (an esem_invariance target is read at the same level). The comparison is then made by orientation transfer: the unstandardized loadings are rotated to the reference and the factor covariances are transformed with the same rotation before standardizing. From the weak level on the non-reference groups have free factor covariances, and standardizing with the diagonal of $Q'\Psi_g Q$ is not rotation-equivariant, so Procrustes alignment of standardized matrices leaves a residual there (about 0.01 to 0.02 on psych::bfi) that the transfer removes; • a list with one element per group, each a list with lambda (unstandardized loadings, items x factors), psi (factor covariance matrix) and theta (residual variances): a solution from elsewhere, compared by the same orientation transfer; • "group1" – align all groups to the first group's loadings within x (within-fit consistency, no external reference); • "canonical" (default) – apply deterministic sign + column order rule (sort columns by $\Sigma\lambda^2$ desc, sign-flip so the largest $\lvert\lambda\rvert$ per column is positive). No external reference required.
level	Only used for esem_invariance objects. One of "configural" (default), "weak", "strong", "strict".
se_method	"approx" (default) returns row-wise diagonal delta-method SEs (ignores within-row covariance between loadings on different factors). "none" returns aligned point estimates only.

Details

Multi-group fits and esem_invariance objects are aligned per group. Item communalities ($\Sigma\lambda^2$), reliability indices (omega, ECV), and model fit are invariant under orthogonal rotation, so alignment affects only the per-loading partition, not substantive conclusions.

Value

An object of class aligned_loadings: a list with one element per group, each containing

loadings aligned standardized loading matrix (items x factors)

se aligned SEs (matching shape) or NULL

Q the orthogonal rotation matrix applied

residual_max, residual_mean max / mean abs residual vs target (only when an external reference was supplied)

The attribute "method" is "canonical", "procrustes" or "transfer".

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data  = HolzingerSwineford1939,
  group = "school"
)


  inv <- esem_invariance(spec)

  # Canonical orientation (no external reference, deterministic)
  aligned <- align_loadings(inv, target = "canonical", level = "configural")
  print(aligned)

  # Within-fit: align all groups to group 1's orientation
  aligned <- align_loadings(inv, target = "group1", level = "configural")


## Not run:
  # Align to Mplus output (requires a Mplus .out file + MplusAutomation)
  mp  <- MplusAutomation::readModels("besem_inv_configural.out")
  tgt <- extract_mplus_loadings(mp)
  aligned <- align_loadings(inv, target = tgt, level = "configural")

## End(Not run)
```

`alignment_check` *Alignment Ratio Check for ICM-CFA Specification*

Description

Tests whether a standard CFA (ICM-CFA) is appropriately specified by computing alignment ratios from the manifest correlation matrix. Based on Mehrvarz & Rouder (2026), who prove that in a correctly specified ICM-CFA, alignment ratios must be invariant across all admissible item quadruples, with the common value equal to the squared latent correlation phi^2.

Usage

```
alignment_check(
  data,
  clusters,
  cfa_fit = NULL,
  is_cor = FALSE,
  min_within_r = 0.05,
  log_sd_thresholds = c(slight = 0.64, moderate = 1.6)
)
```

Arguments

`data`	A `data.frame` or matrix of observed scores. May also be a correlation matrix (set `is_cor = TRUE`).
`clusters`	A named character vector mapping item names to factor names. Example: `c(y1 = "F1", y2 = "F1", y3 = "F2", y4 = "F2")`. Alternatively, a named list: `list(F1 = c("y1","y2"), F2 = c("y3","y4"))`.
`cfa_fit`	Optional. A fitted `lavaan` CFA object. When supplied, CFA-estimated latent correlations are compared against the alignment-implied phi to quantify inflation.
`is_cor`	Logical. Is `data` already a correlation matrix? Default `FALSE`.
`min_within_r`	Numeric. Alignment ratios whose denominator contains a within-cluster correlation below this value are excluded (near-zero within-cluster correlations make ratios numerically unstable). Default `0.05`.
`log_sd_thresholds`	Named numeric vector of log-scale SD cutoffs for the slight / moderate / high misalignment classification. Defaults to the empirical terciles reported by Mehrvarz & Rouder (2026, p.24): `c(slight = 0.64, moderate = 1.6)`. Values of $s = \mathrm{sd}(\log Q)$ below `slight` indicate an ICM-CFA approximately consistent with the data; above `moderate` indicate substantial misspecification.

Details

Interpreting the log-scale dispersion (sd of log Q):

$s = \mathrm{sd}(\log Q)$ is the primary diagnostic. The log scale is natural because alignment ratios are multiplicative objects. Mehrvarz & Rouder (2026, p.24) report empirical terciles of s from their bifactor misalignment simulation:

- **s < 0.64** – Slight misalignment. ICM-CFA is approximately consistent with the data; latent correlations are roughly trustworthy.
- **s 0.64-1.6** – Moderate misalignment. ICM-CFA is partially misspecified; interpret latent correlations with caution.
- **s > 1.6** – High misalignment (top tercile of the paper's simulation, up to ~4.4). ICM-CFA is substantially misspecified and latent correlations are likely inflated. Consider ESEM.

These cutpoints are the 33rd and 66th percentiles of s observed by Mehrvarz & Rouder across a broad range of simulated loading configurations (with $\mathrm{sd}(\log \kappa_j) \in [0, 3]$), not hard decision boundaries; adjust `log_sd_thresholds` if your application calls for stricter or looser cutoffs.

Types of misspecification detected:

- **Misassignment**: items are assigned to the wrong cluster. Mehrvarz & Rouder prove this necessarily inflates phi. The alignment ratios split into three plateaus at phi^2, 1, and 1/phi^2.
- **Misalignment**: cluster assignments are correct but cross-loadings exist (or the proportionality constraint is violated in a bifactor-like DGP). The ratios are dispersed rather than invariant.

Number of alignment ratios per pair:

Mehrvarz & Rouder (2026, p.14) count one alignment ratio per admissible index quadruple: $n_Q = \binom{m_A}{2}\binom{m_B}{2}$. Because each unordered quadruple $\{\ell, \ell'\} \times \{k, k'\}$ admits two between-cluster

matchings ($(\ell, k)/(\ell', k')$ and $(\ell, k')/(\ell', k)$), both of which equal φ^2 under correctly specified ICM-CFA but diverge under misspecification, this function records both matchings per quadruple – yielding $2\binom{m_A}{2}\binom{m_B}{2}$ ratios in `all_ratios` – to maximise the information available for the dispersion diagnostic.

Value

An object of class `"alignment_check"` (a list) containing:

`pair_results` A `data.frame` with one row per factor pair, giving: geometric mean of alignment ratios, log-scale SD, implied phi, CFA-estimated phi (if supplied), inflation percentage, and verdict.

`all_ratios` A named list of raw alignment ratio vectors, one element per factor pair.

`recommendation` Character string: overall recommendation.

`cor_matrix` The manifest correlation matrix used.

`clusters` Resolved cluster assignments (named list).

`call` The matched call.

What are alignment ratios?

For any two items ℓ, ℓ' in cluster A and two items k, k' in cluster B, the alignment ratio is:

$$Q(\ell, \ell', k, k') = \frac{r_{\ell k} \cdot r_{\ell' k'}}{r_{\ell \ell'} \cdot r_{k k'}}$$

Under a correctly specified ICM-CFA, all such ratios equal phi^2 and lie in $[0, 1]$ (Mehrvarz & Rouder, 2026, Eq. 5). Dispersion in the ratios – measured by their log-scale standard deviation $s = \mathrm{sd}(\log Q)$ – signals misspecification: either misassignment (items in the wrong cluster) or misalignment (cross-loadings exist but are fixed to zero).

References

Mehrvarz, M., & Rouder, J. N. (2026). The geometry and brittleness of latent correlations in confirmatory factor analysis.

See Also

`esem` for the recommended follow-up under moderate or high misalignment.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]

# Vector form of cluster assignment
clusters <- c(
  x1 = "Visual",  x2 = "Visual",  x3 = "Visual",
  x4 = "Textual", x5 = "Textual", x6 = "Textual",
  x7 = "Speed",   x8 = "Speed",   x9 = "Speed"
)
```

```
# Run alignment check (data only)
check <- alignment_check(d, clusters)
print(check)

# Run with a fitted CFA to quantify inflation
cfa_model <- "
  Visual  =~ x1 + x2 + x3
  Textual =~ x4 + x5 + x6
  Speed   =~ x7 + x8 + x9
"
cfa_fit <- lavaan::cfa(cfa_model, data = d, std.lv = TRUE)
check   <- alignment_check(d, clusters, cfa_fit = cfa_fit)
print(check)
```

---

besem *Bifactor Exploratory Structural Equation Modeling (B-ESEM)*

Description

Fits a Bifactor ESEM model – a general factor (G) loading on all indicators plus domain-specific factors each targeting a subset of indicators. All factors are orthogonal (uncorrelated), matching the Mplus B-ESEM specification on **continuous** data via lavaan's `efa()` block. With `ordered` indicators, the call routes to `besem_ordered` (default custom DWLS/WLSMV path; see ?`besem_ordered`).

Usage

```
besem(
  data,
  specific_factors,
  indicators = NULL,
  g_name = "G",
  estimator = "MLR",
  std.lv = TRUE,
  ordered = NULL,
  group = NULL,
  group_equal = NULL,
  missing = "listwise",
  n_starts = 30L,
  ...
)
```

Arguments

`data`	A `data.frame` of observed indicators.

`specific_factors`
: A named list mapping specific factor names to their primary indicator names. The general factor G is added automatically. Example: `list(EX = c("y1","y2"), MD = c("y3","y4"), CI = c("y5","y6"))`.

`indicators`
: Optional character vector of all indicator names. If `NULL` (default), all items from `specific_factors` are used.

`g_name`
: Character. Name for the general factor. Default `"G"`.

`estimator`
: Character. Default `"MLR"` (ML with Huber-White robust SEs and Satorra-Bentler scaled chi-square).

`std.lv`
: Logical. Fix factor variances to 1. Default `TRUE`.

`ordered`
: Character vector of ordinal item names. When non-`NULL`, routes to `besem_ordered`. For Mplus-aligned ordered B-ESEM use `method = "rotation"` there (default in `run_comparison`).

`group`
: Character. Grouping variable for multi-group B-ESEM.

`group_equal`
: Character vector of lavaan equality constraints. With `"loadings"` and a `group`, the common loading matrix is rotated without row scaling, as in Mplus (see "Rotation across groups" in `esem_invariance`).

`missing`
: Character. Missing data handling for **continuous** B-ESEM. Default `"listwise"`. When `ordered` is set, passed to `besem_ordered` (default `"listwise"` there; the pipeline passes `"pairwise"` from `specify_model` when used via `run_comparison`).

`n_starts`
: Integer. Number of random orthogonal starting matrices for the target rotation (forwarded as `rstarts` to lavaan's rotation engine). Multiple random starts help escape local optima of the rotation criterion, particularly for bifactor models with many specific factors. Default `30L` (matches Mplus).

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

Details

What makes B-ESEM different from ESEM:

	ESEM	B-ESEM
Factor structure	k oblique specific factors	1 general + k orthogonal specific
Factor correlations	Freely estimated	All fixed to zero (orthogonal)
Cross-loadings	Estimated via rotation	Estimated via orthogonal target rotation
G factor	None	Loads freely on all items
Rotation	Oblique target/geomin	Orthogonal target

Orthogonality:

B-ESEM uses orthogonal target rotation (`"targetT"` in lavaan), which constrains all factors to be uncorrelated. This means:

- G is uncorrelated with EX, MD, CI
- EX, MD, CI are uncorrelated with each other

This matches Mplus `ROTATION = TARGET (orthogonal)`.

Interpreting results:

- **G loadings**: variance shared across all items regardless of domain
- **Specific loadings**: domain-specific variance after accounting for G
- **omega_h (omega hierarchical)**: reliability of G (use `psych::omega()`)
- **omega_s (omega specific)**: reliability of each specific factor

Target matrix structure:

```
      G   EX   MD   CI
y1    1    1    0    0   <- G free on all; EX primary; MD/CI targeted to 0
y2    1    1    0    0
y6    1    0    1    0   <- MD item
y9    1    0    0    1   <- CI item
```

Value

An object of class `c("besem_fit", "esem_fit")` with the same structure as `esem`, plus:

`g_name` Name of the general factor.

`specific_factors` Named list of specific factor assignments.

Mplus target syntax

```
Mplus continuous / ordered target syntax:

ROTATION = TARGET (orthogonal);
MODEL:
  G BY batEX1-batCI5 (*1);
  EX BY batEX1~1 ... batMD1~0 ... (*1);
  MD BY batEX1~0 ... batMD1~1 ... (*1);
  CI BY batEX1~0 ... batCI1~1 ... (*1);
```

Estimator paths

Continuous ESEM / B-ESEM `esem` and `besem` use `lavaan::cfa()` with a native `efa()` block (integrated MLR estimation and rotation).

Ordered ESEM `esem(ordered = ...)` calls `esem_ordered`. Default `method = "lavaan"` (lavaan WLSMV, `efa()` block, post-hoc rotation). `method = "rotation"` uses a custom DWLS pipeline (polychoric correlations + **GPArotation**).

Ordered B-ESEM `besem(ordered = ...)` calls `besem_ordered`. Default `method = "rotation"` (custom DWLS/WLSMV + orthogonal `targetT`; this is what `run_comparison` uses for ordered data). `method = "set-esem"` fits a lavaan WLSMV bifactor **CFA** with non-primary specific loadings fixed at zero; it does **not** match Mplus B-ESEM loadings.

Multi-group invariance `esem_invariance` uses lavaan multi-group `efa()` models plus explicit syntax patches for ordered B-ESEM. See that help page for scope limits.

Missing-data defaults

Defaults differ by entry point; pass `missing` explicitly when fitting ESEM and B-ESEM separately on the same ordered dataset.

- `specify_model` / `run_comparison`: `"pairwise"` for ordered data, `"listwise"` for continuous.
- `esem_ordered`: `"pairwise"`.
- `besem_ordered`: `"listwise"` (pipeline still passes `missing` from the spec when used via `run_comparison`).
- `esem` / `besem` (continuous): `"listwise"`.

The lavaan_fit slot on custom WLSMV fits

For `besem_ordered(method = "rotation")` and `esem_ordered(method = "rotation")`, `$lavaan_fit` is often an **auxiliary** one-factor WLSMV CFA used only to extract DWLS weight matrices—not the fitted ESEM/B-ESEM model. Use `std_loadings`, `parameters`, and `fitMeasures(x)` on the `esem_fit` wrapper; `fitMeasures(x)` reads `wlsmv_stats` when present. Do not interpret `summary(x$lavaan_fit)`, `modindices(x)`, or `coef(x)` as the rotated solution unless you know the fit used the lavaan `efa()` path (`method = "lavaan"` or `method = "set-esem"` for B-ESEM).

See Also

`esem` for standard oblique ESEM, `besem_ordered` for ordinal B-ESEM, `make_bifactor_target` for the target matrix, `generate_mplus_besem_syntax` for Mplus comparison.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]


fit_b <- besem(
  data = d,
  specific_factors = list(
    Visual  = c("x1", "x2", "x3"),
    Textual = c("x4", "x5", "x6"),
    Speed   = c("x7", "x8", "x9")
  ),
  n_starts = 5L
)

summary(fit_b, fit.measures = TRUE, standardized = TRUE)
std_loadings(fit_b)        # rows = items, cols = G + specific factors
factor_correlations(fit_b) # should all be ~0 (orthogonal)

# Compare B-ESEM vs standard ESEM
fit_esem <- esem(d, nfactors = 3)
lavaan::fitMeasures(lavaan_fit(fit_b),    c("cfi", "rmsea", "aic"))
lavaan::fitMeasures(lavaan_fit(fit_esem), c("cfi", "rmsea", "aic"))
```

besem_ordered *Bifactor ESEM for Ordinal Data*

Description

Fits B-ESEM on ordinal indicators. Called from besem when ordered is set. For Mplus-aligned loadings and fit indices use method = "rotation" (default; also used by run_comparison).

Usage

```
besem_ordered(
  data,
  specific_factors,
  indicators = NULL,
  g_name = "G",
  method = c("rotation", "set-esem"),
  n_starts = 30L,
  r_obs_override = NULL,
  group = NULL,
  group_equal = NULL,
  missing = "listwise",
  std.lv = TRUE,
  ...
)
```

Arguments

data	A data.frame of observed ordered indicators.
specific_factors	Named list of specific factor -> item assignments.
indicators	Character vector of all indicator names. If NULL, derived from specific_factors.
g_name	Character. General factor name. Default "G".
method	Character. "rotation" (default): custom DWLS/WLSMV + orthogonal targetT rotation (Mplus-aligned B-ESEM). "set-esem": lavaan WLSMV **bifactor CFA** with non-primary specific loadings fixed at 0* and starts from polychoric EFA—does **not** match Mplus B-ESEM loadings (restricted model; use only for debugging or when you want zero specific-factor cross-loadings).
n_starts	Integer. Random rotation starts. Default 30L (matches Mplus).
r_obs_override	Optional observed correlation matrix to fit instead of lavaan's polychoric correlations. The weight matrix still comes from lavaan, so a matrix that differs from lavaan's by more than 0.001 triggers a warning.
group	Character. Grouping variable for multi-group models.
group_equal	Character vector of lavaan equality constraints.

missing	Character. Missing data handling. Default "listwise". run_comparison passes "pairwise" from specify_model when fitting via the pipeline. Set explicitly when comparing with esem_ordered (default "pairwise").
std.lv	Logical. Default TRUE.
...	Additional arguments passed to lavaan::cfa().

Value

An object of class c("besem_fit_ordered", "besem_fit", "esem_fit"). For method = "rotation", lavaan_fit is auxiliary (one-factor CFA); use std_rotated_loadings, wlsmv_stats, std_loadings, and fitMeasures(x); communalities holds each item's communality, and a value above 1 (a Heywood case, negative residual variance) gives a warning. For method = "set-esem", lavaan_fit is the fitted bifactor CFA.

Method "set-esem" vs "rotation"

rotation estimates cross-loadings (targeted toward zero), then rotates— same estimand as Mplus B-ESEM WLSMV. set-esem **fixes** non-primary specific loadings at zero in lavaan; G and primary loadings are re-optimized under that harder constraint, so loadings and fit differ from Mplus.

Estimator paths

Continuous ESEM / B-ESEM esem and besem use lavaan::cfa() with a native efa() block (integrated MLR estimation and rotation).

Ordered ESEM esem(ordered = ...) calls esem_ordered. Default method = "lavaan" (lavaan WLSMV, efa() block, post-hoc rotation). method = "rotation" uses a custom DWLS pipeline (polychoric correlations + **GPArotation**).

Ordered B-ESEM besem(ordered = ...) calls besem_ordered. Default method = "rotation" (custom DWLS/WLSMV + orthogonal targetT; this is what run_comparison uses for ordered data). method = "set-esem" fits a lavaan WLSMV bifactor **CFA** with non-primary specific loadings fixed at zero; it does **not** match Mplus B-ESEM loadings.

Multi-group invariance esem_invariance uses lavaan multi-group efa() models plus explicit syntax patches for ordered B-ESEM. See that help page for scope limits.

Missing-data defaults

Defaults differ by entry point; pass missing explicitly when fitting ESEM and B-ESEM separately on the same ordered dataset.

- specify_model / run_comparison: "pairwise" for ordered data, "listwise" for continuous.
- esem_ordered: "pairwise".
- besem_ordered: "listwise" (pipeline still passes missing from the spec when used via run_comparison).
- esem / besem (continuous): "listwise".

The lavaan_fit slot on custom WLSMV fits

For `besem_ordered(method = "rotation")` and `esem_ordered(method = "rotation")`, `$lavaan_fit` is often an **auxiliary** one-factor WLSMV CFA used only to extract DWLS weight matrices—not the fitted ESEM/B-ESEM model. Use `std_loadings`, `parameters`, and `fitMeasures(x)` on the `esem_fit` wrapper; `fitMeasures(x)` reads `wlsmv_stats` when present. Do not interpret `summary(x$lavaan_fit)`, `modindices(x)`, or `coef(x)` as the rotated solution unless you know the fit used the lavaan `efa()` path (`method = "lavaan"` or `method = "set-esem"` for B-ESEM).

See Also

`esem_ordered`, `besem`, `run_comparison`

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

# Derive ordered (5-category Likert) versions of the 9 continuous items
items <- paste0("x", 1:9)
ord <- as.data.frame(lapply(HolzingerSwineford1939[, items], function(v) {
  as.integer(cut(v, breaks = quantile(v, probs = seq(0, 1, 0.2)),
                 include.lowest = TRUE))
}))
names(ord) <- items


fit_b_ord <- besem_ordered(
  data = ord,
  specific_factors = list(
    Visual  = c("x1", "x2", "x3"),
    Textual = c("x4", "x5", "x6"),
    Speed   = c("x7", "x8", "x9")
  ),
  n_starts = 5L
)
summary(fit_b_ord, fit.measures = TRUE, standardized = TRUE)
```

---

bifactory_template *Open or Copy the bifactory Analysis Template*

Description

Opens the shipped analysis template in your editor, or copies it to a destination of your choice. The template walks through the full bifactory pipeline (CFA / ESEM / B-ESEM comparison, reliability, ESEM-within-CFA, factor scores, and optional multi-group invariance) using `psych::bfi` as the demo dataset.

Usage

```
bifactory_template(to = NULL, overwrite = FALSE)
```

Arguments

to	Optional file path. If supplied, the template is copied there (with overwrite protection). If NULL (default), the template is opened in the editor via file.edit.
overwrite	Logical. Overwrite an existing file at to? Default FALSE.

Value

Invisibly returns the path to the template (or the destination when to is supplied).

Examples

```
# Find the template path
system.file("templates", "template.R", package = "bifactory")

# Copy the template to a file (here a temporary one)
dest <- file.path(tempdir(), "my_analysis.R")
bifactory_template(to = dest, overwrite = TRUE)

## Not run:
# Open the template directly in your editor (interactive session only)
bifactory_template()

## End(Not run)
```

---

chisq_decomp *Per-Group Chi-Square Decomposition for Invariance Fits*

Description

Extracts the per-group contribution to the WLSMV (or ML) chi-square at each invariance level of an esem_invariance object. Optionally compares against Mplus's *Chi-Square Contribution From Each Group* table, which is useful for pinpointing whether a basin difference between R and Mplus is driven by a single group or distributed across all of them.

Usage

```
chisq_decomp(x, mplus_dir = NULL, levels = NULL)
```

Arguments

x	An esem_invariance object.
mplus_dir	Optional directory containing Mplus output files named besem_inv_<level>.out. When supplied, the result includes chisq_M, mplus_group, and delta = chisq_R - chisq_M.
levels	Character vector of levels to include. Defaults to all levels present in x$models.

Details

lavaan's stat.group is the raw (unscaled) per-group chi-square; Mplus prints the *scaled* per-group contribution that sums to the scaled total. To compare on the same scale, R per-group values are rescaled by the global ratio chisq.scaled / chisq from fitMeasures.

Value

A data frame of class "chisq_decomp" with columns level, group, chisq_R (rescaled to scaled scale), and when mplus_dir is supplied, mplus_group, chisq_M, delta.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data  = HolzingerSwineford1939,
  group = "school"
)


inv <- esem_invariance(spec)
chisq_decomp(inv)


## Not run:
# Supply a folder of Mplus .out files to add side-by-side deltas.
chisq_decomp(inv, mplus_dir = "validation/_bfi_g4_mplus_inv")

## End(Not run)
```

---

coef.esem_fit *Extract Parameter Estimates from an esem_fit*

Description

Forwards to lavaan on lavaan_fit(x). On custom WLSMV rotation paths, use parameters or std_loadings instead; see The lavaan_fit slot on custom WLSMV fits.

Usage

```
## S3 method for class 'esem_fit'
coef(object, standardized = FALSE, ...)
```

Arguments

object	An esem_fit object.
standardized	Logical. Return standardized estimates? Default FALSE.
...	Passed to lavaan::parameterEstimates().

Value

A data.frame of parameter estimates from lavaan::parameterEstimates(), or the standardized solution from lavaan::standardizedsolution() when standardized = TRUE.

---

compare_ewc *Compare Pipeline Results with an EWC Model*

Description

Appends fit indices from a fit_ewc result to the pipeline comparison table, producing a unified data frame with CFA, ESEM, B-ESEM, and EWC columns side by side.

Usage

```
compare_ewc(results, ewc)
```

Arguments

results	An esem_comparison_pipeline from run_comparison.
ewc	An ewc_fit from fit_ewc.

Value

An object of class c("ewc_comparison","data.frame"). Print with print() for a formatted table.

See Also

fit_ewc

`compare_loadings` *Compare R and Mplus Standardised Loadings*

Description

Builds a long-format data frame with STDYX loadings, standard errors, z-scores, p-values, and R − Mplus differences for every loading (primary and cross) across CFA, ESEM, and B-ESEM. Mplus columns are `NA` when no Mplus results are present in the pipeline object.

Usage

```
compare_loadings(results)
```

Arguments

`results`	An `esem_comparison_pipeline` object from `run_comparison`.

Value

A data frame with columns:

`model` CFA, ESEM, or BESEM.

`factor` Factor name (lowercase).

`item` Item name (lowercase).

`loading_type` `"primary"` or `"cross"`.

`R_std`, `R_se`, `R_z`, `R_p` R estimates.

`Mplus_std`, `Mplus_se`, `Mplus_z`, `Mplus_p` Mplus estimates (`NA` if unavailable).

`diff_std` R − Mplus standardised loading difference (`NA` if unavailable).

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data    = HolzingerSwineford1939,
  label   = "Holzinger-Swineford"
)


results <- run_comparison(spec, n_starts = 5L)
lc <- compare_loadings(results)
head(lc)

# Primary loadings only (Mplus_* columns are NA without a Mplus run)
```

```
lc[lc$loading_type == "primary", ]
```

`compute_indices` *Compute Reliability Indices for CFA, ESEM, and B-ESEM*

Description

Computes McDonald's omega reliability indices for all three models in a pipeline result, following the bifactor reporting framework recommended by Morin, Arens & Marsh (2016) and Rodriguez, Reise & Haviland (2016).

Usage

```
compute_indices(results)
```

Arguments

`results`	An `esem_comparison_pipeline` object from `run_comparison`.

Details

For the **total composite** (all items):

- `omega_total` – total reliability (rotation-invariant for orthogonal models).
- `omega_H` – hierarchical omega: G factor's contribution to total-score reliability (B-ESEM only).
- `ECV` – explained common variance: G's share of all common variance (B-ESEM only).
- `H(G)` – construct replicability of G (B-ESEM only).

For each **subscale** (items of specific factor F_s):

- `omega_S` – specific factor's contribution to subscale reliability: $(\sum_{i \in s} |\lambda_{s,i}|)^2 / [(\sum_{i \in s} |\lambda_{G,i}|)^2 + (\sum_{i \in s} |\lambda_{s,i}|)^2 + \sum_{i \in s} \psi_i]$. G is not partialled out; its variance is treated as an additional source of error (Morin, 2023; Morin et al., 2020). `omega_H(G|s)` is the mirror image with the specific factor treated as error.
- `omega_sub` – total reliability of the subscale sum score (G + specific factor combined).
- `omega_H_sub` – G's contribution to subscale reliability.
- `ECV_s` – G's share of common variance within the subscale.
- `H(Fs)` – construct replicability of the specific factor.

Every composite sum uses absolute standardized loadings, the convention of Morin, Arens and Marsh (2016): each item is scored in the direction of its loading, so reverse-worded items and the orientation of a factor do not change any index. Residual variances are taken from the model-implied item variances (for correlated factors, one minus the diagonal of $\Lambda\Phi\Lambda'$).

Value

An object of class `"reliability_indices"` – a named list with elements `cfa`, `esem`, `besem`, each containing the computed indices for that model, plus `alpha` (Cronbach's alpha per subscale and for G). Pass to `print()` for a formatted table.

References

McDonald, R. P. (1999). *Test theory: A unified treatment*. Erlbaum.

Rodriguez, A., Reise, S. P., & Haviland, M. G. (2016). Evaluating bifactor models: Calculating and interpreting statistical indices. *Psychological Methods*, *21*(2), 137-150.

Morin, A. J. S., Arens, A. K., & Marsh, H. W. (2016). A bifactor exploratory structural equation modeling framework for the identification of distinct sources of construct-relevant psychometric multidimensionality. *Structural Equation Modeling*, *23*(1), 116-139.

Morin, A. J. S., Myers, N. D., & Lee, S. (2020). Modern factor analytic techniques: Bifactor models, exploratory structural equation modeling (ESEM) and bifactor-ESEM. In G. Tenenbaum & R. C. Eklund (Eds.), *Handbook of Sport Psychology* (4th ed., pp. 1044-1073). Wiley.

Morin, A. J. S. (2023). Exploratory structural equation modeling. In R. H. Hoyle (Ed.), *Handbook of Structural Equation Modeling* (2nd ed., pp. 503-524). Guilford.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data    = HolzingerSwineford1939,
  label   = "Holzinger-Swineford"
)


results <- run_comparison(spec, n_starts = 5L)
indices <- compute_indices(results)
print(indices)
```

---

compute_omega *Deprecated: use* compute_indices *instead*

Description

Deprecated: use `compute_indices` instead

Usage

```
compute_omega(results)
```

Arguments

results	An esem_comparison_pipeline object.

Value

A reliability_indices object.

---

esem *Exploratory Structural Equation Modeling*

Description

Fits ESEM using **lavaan**'s native efa() block. For **continuous** data (default), estimation and rotation are integrated in lavaan's ML optimiser—the same single-step approach as Mplus (*1) syntax. For **ordered** indicators, supply ordered and the call is routed to esem_ordered (set-ESEM / WLSMV; not identical to continuous esem()).

Usage

```
esem(
  data,
  nfactors,
  indicators = NULL,
  rotation = "geomin",
  target = NULL,
  estimator = "MLR",
  std.lv = TRUE,
  ordered = NULL,
  group = NULL,
  group_equal = NULL,
  missing = "listwise",
  factor_names = NULL,
  rotation_args = list(),
  heywood_fix = TRUE,
  ...
)
```

Arguments

data	A data.frame containing the observed indicators (and optionally a grouping variable).
nfactors	Integer. Number of latent factors.
indicators	Character vector of item names to include. If NULL (default), all columns of data except group are used.
rotation	Character. Rotation criterion. Passed to lavaan::cfa(rotation = ...). Common choices:

- `"geomin"` (default) – oblique geomin, the Mplus default
- `"target"` – requires `target` matrix
- `"varimax"` – orthogonal varimax
- `"oblimin"` – oblique oblimin
- `"none"` – no rotation (confirmatory EFA)

The full list of lavaan-supported rotations is in `?lavaan::efaRotate`.

`target` — A numeric matrix (items x factors) for target rotation. Create with `make_target`. Required when `rotation = "target"`.

`estimator` — Character. lavaan estimator. Default `"MLR"` (ML with Huber-White robust SEs and Satorra-Bentler scaled chi-square). Use `"WLSMV"` for ordered indicators.

`std.lv` — Logical. Fix factor variances to 1 for identification? Default `TRUE` (recommended for ESEM).

`ordered` — Character vector of ordinal item names. When non-`NULL`, routes to `esem_ordered` (WLSMV). Default `method` there is `"lavaan"`; see `?esem_ordered` for the custom `"rotation"` path.

`group` — Character. Name of a grouping variable in `data` for multi-group ESEM (configural by default).

`group_equal` — Character vector of lavaan parameter labels to constrain equal across groups, e.g. `"loadings"` (metric) or `c("loadings", "intercepts")` (scalar). With `"loadings"` and a `group`, the common loading matrix is rotated without row scaling, as in Mplus (see "Rotation across groups" in `esem_invariance`).

`missing` — Character. Missing data handling for **continuous** fits. Default `"listwise"`; use `"fiml"` for full-information ML. When `ordered` is set, passed to `esem_ordered` (default `"pairwise"` there unless you override).

`factor_names` — Optional character vector of length `nfactors`. Defaults to `F1, F2, ...`.

`rotation_args` — Named list of extra arguments passed to the rotation function (e.g. `list(geomin.epsilon = 0.001)`). When `rotation = "geomin"` (or any geomin variant) and `geomin.epsilon` is not supplied, it defaults to `0.0001` / `0.001` / `0.01` for 2 / 3 / 4+ factors respectively, matching Mplus's defaults.

`heywood_fix` — Logical. Retry rotation with Cholesky unrotation if a standardised loading exceeds 1 (single-group continuous fits only). Default `TRUE`. If correction runs, `std_loadings` may differ from `lavaan::standardizedsolution(lavaan_fit(x))`; see Heywood fix and loadings.

`...` — Additional arguments passed to `lavaan::cfa()`.

Details

Mplus equivalence:

Mplus syntax	bifactory equivalent
`F1-F3 BY y1-y15 (*1);`	`esem(data, nfactors = 3)`
`(*1)` with geomin (default)	`rotation = "geomin"`
`(*1)` with target	`rotation = "target", target = tgt`
`GROUPING = g;`	`group = "g"`

Metric invariance	`group_equal = "loadings"`
Scalar invariance	`group_equal = c("loadings", "intercepts")`

Why not automate cross-loadings from a CFA?:

Adding cross-loadings stepwise from modification indices is a different (exploratory CFA) approach and is not recommended: it capitalises on chance, inflates Type I error, and produces a different model on every dataset. ESEM instead estimates *all* cross-loadings simultaneously, with rotation acting as a mathematical penalty for complexity – giving a reproducible, theory-neutral solution.

Value

An object of class `"esem_fit"` containing:

`lavaan_fit` The **lavaan** fit object for continuous ESEM. For ordered / custom WLSMV paths see `esem_ordered` and The lavaan_fit slot on custom WLSMV fits.

`syntax` The lavaan model string that was estimated.

`nfactors` Number of factors.

`rotation` Rotation method used.

`factor_names` Factor names.

`indicators` Item names included in the model.

`call` The matched call.

How this differs from a two-stage EFA + CFA workaround

Many R implementations run EFA first, extract the loading matrix, then paste it into a CFA as starting values. That is an approximation. **bifactory** instead uses **lavaan**'s `efa()` block syntax (available since lavaan 0.6-12), which estimates the rotation and the SEM parameters simultaneously – exactly as Mplus does with the `(*1)` syntax.

Estimator paths

Continuous ESEM / B-ESEM `esem` and `besem` use `lavaan::cfa()` with a native `efa()` block (integrated MLR estimation and rotation).

Ordered ESEM `esem(ordered = ...)` calls `esem_ordered`. Default `method = "lavaan"` (lavaan WLSMV, `efa()` block, post-hoc rotation). `method = "rotation"` uses a custom DWLS pipeline (polychoric correlations + **GPArotation**).

Ordered B-ESEM `besem(ordered = ...)` calls `besem_ordered`. Default `method = "rotation"` (custom DWLS/WLSMV + orthogonal `targetT`; this is what `run_comparison` uses for ordered data). `method = "set-esem"` fits a lavaan WLSMV bifactor **CFA** with non-primary specific loadings fixed at zero; it does **not** match Mplus B-ESEM loadings.

Multi-group invariance `esem_invariance` uses lavaan multi-group `efa()` models plus explicit syntax patches for ordered B-ESEM. See that help page for scope limits.

Missing-data defaults

Defaults differ by entry point; pass `missing` explicitly when fitting ESEM and B-ESEM separately on the same ordered dataset.

- `specify_model` / `run_comparison`: `"pairwise"` for ordered data, `"listwise"` for continuous.
- `esem_ordered`: `"pairwise"`.
- `besem_ordered`: `"listwise"` (pipeline still passes `missing` from the spec when used via `run_comparison`).
- `esem` / `besem` (continuous): `"listwise"`.

Heywood fix and loadings

When `heywood_fix = TRUE` (default on single-group `esem` and `esem_ordered`), `std_loadings` may reflect a post-hoc rotation correction while `lavaan_fit` still holds the pre-correction lavaan solution. Prefer `std_loadings(x)` for reported loadings in that case.

See Also

`make_target` for target matrices, `esem_ordered` for ordinal data, `esem_compare` for ESEM vs CFA comparison, `std_loadings` for the standardised loading matrix.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]

# Basic ESEM: 3 factors, geomin rotation (Mplus default)
fit <- esem(d, nfactors = 3)
round(std_loadings(fit), 2)
lavaan::fitMeasures(lavaan_fit(fit), c("cfi", "tli", "rmsea", "srmr"))

# Named factors
fit2 <- esem(d, nfactors = 3,
             factor_names = c("Visual", "Textual", "Speed"))


# Target rotation
tgt <- make_target(list(Vis = 1:3, Txt = 4:6, Spd = 7:9), nitems = 9)
fit3 <- esem(d, nfactors = 3, rotation = "target", target = tgt)

# Multi-group configural ESEM
fit_mg <- esem(HolzingerSwineford1939, nfactors = 3,
               indicators = paste0("x", 1:9), group = "sex")
```

esem_compare *Compare ESEM Against a Standard CFA*

Description

Fits a user-specified CFA model on the same data and compares it against an esem_fit object using fit indices and, where applicable, a chi-square difference test (when models are nested).

Usage

```
esem_compare(esem_model, cfa_model, data = NULL, estimator = NULL, ...)
```

Arguments

esem_model	An esem_fit object from esem.
cfa_model	A lavaan model string for the comparison CFA. All items must be the same as in the ESEM.
data	A data.frame used to fit the CFA. If NULL (default), the data are re-extracted from the esem_model lavaan object.
estimator	Estimator for the CFA model. Defaults to the same estimator used in esem_model.
...	Additional arguments passed to lavaan::cfa().

Value

A list of class "esem_comparison" with:

fit_table A data.frame of fit indices for both models.

esem_fit The original esem_fit object.

cfa_fit The fitted lavaan CFA object.

lavtest Output of lavaan::lavTestLRT() if models are nested, else NULL.

See Also

esem

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]


# Fit ESEM
esem_result <- esem(d, nfactors = 3)

# Define comparison CFA (no cross-loadings)
cfa_model <- "
  Visual  =~ x1 + x2 + x3
```

```
  Textual =~ x4 + x5 + x6
  Speed   =~ x7 + x8 + x9
"
comparison <- esem_compare(esem_result, cfa_model, data = d)
print(comparison)
```

`esem_invariance` *Measurement Invariance Testing for ESEM Models*

Description

Tests configural, weak (metric), strong (scalar), and strict invariance for an ESEM model across groups, optionally followed by latent variance/covariance and latent mean invariance (`through`). Returns a formatted table of fit indices and chi-square difference tests, analogous to Mplus's multi-group output.

Usage

```
esem_invariance(
  spec,
  model = c("esem", "besem"),
  missing = NULL,
  verbose = TRUE,
  through = c("strict", "varcov", "means"),
  cores = 2L,
  ...
)
```

Arguments

`spec`	An `esem_spec` object from `specify_model` that includes a `group` variable. If `spec$group` is `NULL`, prints a message and returns `invisible(NULL)` (not an error).
`model`	Character. Which model type to test: `"esem"` (default) for standard ESEM, or `"besem"` for Bifactor ESEM. Ordered B-ESEM uses lavaan multi-group `efa()` with explicit syntax patches (orthogonal rotation, Theta identification, threshold labels); see **Scope** below.
`missing`	Character. Missing data handling passed to `lavaan::cfa()`. Default `NULL` uses `spec$missing` from `specify_model` (`"pairwise"` for ordered data, `"listwise"` for continuous).
`verbose`	Logical. Print progress messages. Default `TRUE`. The `Fitting <level> ...` line is shown in bold blue where the console supports ANSI colour (RStudio, or a terminal whose `TERM` is not "dumb"); `options(bifactory.ansi_colour = FALSE)` turns the colour off, `TRUE` forces it.

through — Character. Last level of the sequence to fit. "strict" (default) stops at the four measurement-invariance levels. "varcov" adds latent variance/covariance invariance (level 5: factor variances fixed to 1 in every group and factor covariances equal across groups, i.e. 0 in every group for B-ESEM). "means" adds latent mean invariance (level 6: factor means fixed to 0 in every group). Levels are cumulative, so "means" fits all six models.

cores — Integer. Worker processes for the two starting-value fits of each level (lavaan's default start and start = "simple"), which are independent and run at the same time when cores >= 2 (default 2, the most that helps). The results are identical; a level then takes as long as its slower start instead of both together (psych::bfi, three groups, six levels: 13 instead of 18 minutes). 1 fits sequentially. The workers load this package and the calling session's library path.

... — Additional arguments passed to the underlying fit function. Do not pass group, group_equal, ordered, or parameterization here – these are managed internally.

Value

An object of class "esem_invariance" containing:

table Data frame with fit indices and D-statistics. print() renders it formatted.

models Named list of fit objects, one per fitted level: configural, weak, strong, strict, and, with through, varcov and means. NULL entries indicate a model that failed to fit.

lrt Named list of lavTestLRT() outputs, one per fitted level after configural (each compared with the level before it).

spec The original model specification.

model Character: "esem" or "besem".

notes Named list, one character vector per level that raised a lavaan warning while being fitted or compared (singular information matrix for the robust test statistic, non-positive-definite latent or residual covariance matrix, negative variance). The warnings are caught, reworded and shown by print() instead of being raised, so a run that concludes does not end with a lavaan warning. Empty when nothing was raised.

fallback_from, fallback_note If B-ESEM configural fails to converge, the result may be from an **ESEM** fallback (no general factor); not comparable df-for-df to Mplus B-ESEM.

Scope and caveats

- **Ordered B-ESEM:** Validated against Mplus on BFI for $G \in \{2, 3, 4\}$ (max $|\Delta\text{CFI}| \leq 0.001$, $|\Delta\text{SRMR}| \leq 0.002$, df match). At $G \geq 5$ with ≥ 18 items a warning is issued; verify externally before reporting.
- **WLSMV SRMR:** The invariance table applies an Mplus-style SRMR denominator correction for ordered models where implemented.
- **B-ESEM configural failure:** After retries, may fall back to ESEM (fallback_from = "besem"). The printed table is structurally different from B-ESEM; do not compare to Mplus B-ESEM output.

- **Empty categories:** Ordered multi-group fits may recode categories absent in one group (see verbose output).

See `system.file("VALIDATION.md", package = "bifactory")` for a summary.

Constraint mapping

Level	Continuous (MLR)	Ordered (WLSMV/Theta)
Configural	free	free
Weak	loadings	loadings
Strong	loadings + intercepts	loadings + thresholds
Strict	+ residuals	+ residuals
Latent var/cov (`through = "varcov"`)	+ factor variances (1) and covariances	+ factor variances (1) and covariances
Latent means (`through = "means"`)	+ factor means (0)	+ factor means (0)

The last two levels follow Morin, Arens and Marsh (2016): the degrees of freedom grow by $k(k+1)/2$ (variances and covariances of k factors) and then by k (means) per non-reference group.

Scaled chi-square difference tests

Simple subtraction of scaled chi-square values is not valid for MLR or WLSMV. `lavaan::lavTestLRT()` is used, which applies the Satorra-Bentler (2001) correction for MLR and a mean-variance-adjusted difference test for WLSMV. WLSMV difference test results may differ numerically from Mplus's `DIFFTEST` procedure.

Identification across groups

At the configural level `std.lv = TRUE` fixes factor variances to 1 in all groups. Under weak invariance lavaan automatically frees factor variances in Group 2+ and keeps them at 1 in Group 1. Under strong invariance factor means are freed in Group 2+ and fixed to 0 in Group 1. The optional latent var/cov level fixes the Group 2+ variances back to 1 and equates the covariances; the latent mean level fixes the Group 2+ means back to 0.

With ordered indicators (theta parameterization) the configural model uses the standard identification in every group: residual variances 1, factor means 0, all thresholds free. From the weak level on, the non-reference residual variances and factor means are free and identified through the equal loadings plus partial threshold equalities (two thresholds of one referent item per factor, one threshold of every other item), which is the Mplus convention. The same scheme is not used at the configural level: with free loadings an item's single equated threshold is its only link across groups, and when that threshold lies near zero in a non-reference group the item's residual variance is empirically unidentified (it ran to several hundred on `psych::bfi`, with the chi-square drifting by 14 to 36 units between programs). The standard identification is the equivalent, well-conditioned model.

Group 1 (the reference group) is the first sorted level of the grouping variable, `spec$group_levels`, which is also the reference group in the Mplus syntax written by `run_mplus_besem_invariance`. The fits pass this order to lavaan as `group.label`; lavaan's own default would be the order in which the groups appear in the data, and a different reference group gives an equivalent model with the same fit but differently oriented loadings and factor covariances.

Rotation across groups

At the configural level each group is rotated on its own, with the rows of its loading matrix scaled by the model-implied item variances, as in Mplus. From the weak level on the loadings are equal across groups and the common loading matrix is rotated once, without row scaling (`rotation.args = list(std.ov = FALSE)` in lavaan), which is what Mplus does. lavaan's default would scale the rows by the residual variances averaged over the groups: the same fit, but a different rotation of it, with standardized loadings 0.03 to 0.09 away from Mplus's on `psych::bfi` (0.44 in one bifactor model) and standard errors up to 0.003 away. Up to 0.6.1 bifactory used that default.

The unscaled rotation works on the loadings in the metric of the first group. `esem_invariance()` passes `group.label` so that this is the first sorted group level, as in the generated Mplus syntax; direct calls of `esem()`, `besem()` or `esem_ordered()` with `group` and `group_equal = "loadings"` follow lavaan's order (first appearance in the data) unless `group.label` is passed through `...`. The agreement with Mplus was checked for target rotation. Mplus does not allow Kaiser row standardization with equal loadings, while lavaan keeps Kaiser weights for varimax and promax.

With oblique target rotation (ESEM) the rotation standard errors use lavaan's delta method (`rotation.se = "delta"`) with `rotation.args$gpa.tol = 1e-6`, which agrees more closely with Mplus at the configural level than lavaan's default `"bordered"`; the tighter tolerance keeps the delta-method standard errors from depending on the random rotation starts. B-ESEM (orthogonal target rotation) keeps `"bordered"`. A `rotation.se` passed through `...` is kept.

Estimator paths

Continuous ESEM / B-ESEM `esem` and `besem` use `lavaan::cfa()` with a native `efa()` block (integrated MLR estimation and rotation).

Ordered ESEM `esem(ordered = ...)` calls `esem_ordered`. Default `method = "lavaan"` (lavaan WLSMV, `efa()` block, post-hoc rotation). `method = "rotation"` uses a custom DWLS pipeline (polychoric correlations + **GPArotation**).

Ordered B-ESEM `besem(ordered = ...)` calls `besem_ordered`. Default `method = "rotation"` (custom DWLS/WLSMV + orthogonal `targetT`; this is what `run_comparison` uses for ordered data). `method = "set-esem"` fits a lavaan WLSMV bifactor **CFA** with non-primary specific loadings fixed at zero; it does **not** match Mplus B-ESEM loadings.

Multi-group invariance `esem_invariance` uses lavaan multi-group `efa()` models plus explicit syntax patches for ordered B-ESEM. See that help page for scope limits.

See Also

`specify_model`, `esem`, `esem_ordered`

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data  = HolzingerSwineford1939,
  group = "school",
```

```
  label = "Holzinger-Swineford"
)


# Continuous ESEM measurement invariance across schools
inv <- esem_invariance(spec)
print(inv)

# Access individual model fits
summary(inv$models$strong, fit.measures = TRUE, standardized = TRUE)
lavaan::lavTestScore(inv$models$strong$lavaan_fit)

# Ordered data (WLSMV) and B-ESEM invariance follow the same pattern:
# add `ordered = TRUE` to specify_model(), then optionally `model = "besem"`.
items <- paste0("x", 1:9)
ord <- as.data.frame(lapply(HolzingerSwineford1939[, items], function(v) {
  as.integer(cut(v, breaks = quantile(v, probs = seq(0, 1, 0.2)),
                 include.lowest = TRUE))
}))
names(ord) <- items
ord$school <- HolzingerSwineford1939$school

spec_ord <- specify_model(
  Visual = c("x1", "x2", "x3"), Textual = c("x4", "x5", "x6"),
  Speed = c("x7", "x8", "x9"),
  data = ord, group = "school", ordered = TRUE
)
inv_besem <- esem_invariance(spec_ord, model = "besem")
print(inv_besem)
```

`esem_ordered` *ESEM for Ordinal (Likert) Data*

Description

Fits ESEM on ordinal indicators (WLSMV / Theta parameterization). Called automatically from `esem` when `ordered` is set.

Usage

```
esem_ordered(
  data,
  nfactors,
  indicators = NULL,
  rotation = "target",
  target = NULL,
  factor_names = NULL,
```

```
  method = c("lavaan", "rotation"),
  n_starts = 100L,
  r_obs_override = NULL,
  std.lv = TRUE,
  missing = "pairwise",
  group = NULL,
  group_equal = NULL,
  heywood_fix = TRUE,
  n_obs = NULL,
  ...
)
```

Arguments

data	A data.frame of observed ordered indicators.
nfactors	Integer. Number of latent factors.
indicators	Character vector of item names. If NULL, all columns of data except group are used.
rotation	Character. Rotation method. Default "target". Supports "target" (oblique), "targetT" (orthogonal), "geomin", "geominT" (orthogonal geomin), "oblimin", "varimax".
target	A target matrix from make_target. Required when rotation contains "target". Convention: NA = free (no penalty), 0 = targeted toward zero.
factor_names	Optional character vector of factor names.
method	Character. "lavaan" (default): lavaan efa() + WLSMV. "rotation": custom DWLS + post-hoc target rotation (see The lavaan_fit slot on custom WLSMV fits).
n_starts	Integer. Number of random rotation starts when method = "rotation". Default 100L.
r_obs_override	Optional observed correlation matrix to fit instead of lavaan's polychoric correlations (for reproducibility / testing). The weight matrix still comes from lavaan, so a matrix that differs from lavaan's by more than 0.001 triggers a warning.
std.lv	Logical. Fix factor variances to 1. Default TRUE.
missing	Character. Missing data handling passed to lavaan. Default "pairwise" (Mplus-style polychoric pairs). Use "listwise" for complete cases. When comparing with besem_ordered on the same data, set missing explicitly on both calls (besem_ordered defaults to "listwise").
group	Character. Grouping variable name for multi-group models.
group_equal	Character vector of lavaan equality constraints. With "loadings" and a group, the common loading matrix is rotated without row scaling, as in Mplus (see "Rotation across groups" in esem_invariance).
heywood_fix	Logical. Retry rotation with Cholesky unrotation when a standardised loading exceeds 1. Default TRUE.
n_obs	Ignored (kept for backward compatibility).
...	Additional arguments passed to lavaan::cfa().

Details

Default (`method = "lavaan"`): lavaan's `efa()` block with WLSMV (unrestricted model, post-hoc rotation, delta-method SEs)—aligned with Mplus `ESTIMATOR = WLSMV; ROTATION = TARGET (oblique)`.

Alternate (`method = "rotation"`): custom DWLS pipeline (polychoric correlations, weight matrix and asymptotic covariance from one auxiliary **lavaan** fit, **GPArotation**, sandwich + rotation Jacobian SEs).

Mplus equivalence:

```
Mplus:
  ANALYSIS:
    ESTIMATOR = WLSMV;
    ROTATION  = TARGET;
    PARAMETERIZATION = THETA;
  MODEL:
    EX MD CI BY item1-item18 (*1);

R (this function):
  esem_ordered(data, nfactors = 3, rotation = "target", target = tgt,
               factor_names = c("EX","MD","CI"))
```

Algorithm (`method = "lavaan"`)**:**

lavaan's `efa()` block fits an unrestricted k-factor model under WLSMV, applies post-hoc rotation, and propagates standard errors through the rotation transformation (Asparouhov & Muthen, 2009).

Value

An object of class `c("esem_fit_ordered", "esem_fit")`. For `method = "lavaan"`, `lavaan_fit` is the fitted ESEM model. For `method = "rotation"`, `lavaan_fit` is an auxiliary one-factor CFA; use `std_rotated_loadings`, `wlsmv_stats`, and `std_loadings`. Slot `estimator` is `"WLSMV"` or `"DWLS"` depending on path. For `method = "rotation"`, `communalities` holds each item's communality; a value above 1 (a Heywood case, negative residual variance) gives a warning.

Estimator paths

Continuous ESEM / B-ESEM `esem` and `besem` use `lavaan::cfa()` with a native `efa()` block (integrated MLR estimation and rotation).

Ordered ESEM `esem(ordered = ...)` calls `esem_ordered`. Default `method = "lavaan"` (lavaan WLSMV, `efa()` block, post-hoc rotation). `method = "rotation"` uses a custom DWLS pipeline (polychoric correlations + **GPArotation**).

Ordered B-ESEM `besem(ordered = ...)` calls `besem_ordered`. Default `method = "rotation"` (custom DWLS/WLSMV + orthogonal `targetT`; this is what `run_comparison` uses for ordered data). `method = "set-esem"` fits a lavaan WLSMV bifactor **CFA** with non-primary specific loadings fixed at zero; it does **not** match Mplus B-ESEM loadings.

Multi-group invariance `esem_invariance` uses lavaan multi-group `efa()` models plus explicit syntax patches for ordered B-ESEM. See that help page for scope limits.

Missing-data defaults

Defaults differ by entry point; pass `missing` explicitly when fitting ESEM and B-ESEM separately on the same ordered dataset.

- `specify_model` / `run_comparison`: `"pairwise"` for ordered data, `"listwise"` for continuous.
- `esem_ordered`: `"pairwise"`.
- `besem_ordered`: `"listwise"` (pipeline still passes `missing` from the spec when used via `run_comparison`).
- `esem` / `besem` (continuous): `"listwise"`.

The lavaan_fit slot on custom WLSMV fits

For `besem_ordered(method = "rotation")` and `esem_ordered(method = "rotation")`, `$lavaan_fit` is often an **auxiliary** one-factor WLSMV CFA used only to extract DWLS weight matrices—not the fitted ESEM/B-ESEM model. Use `std_loadings`, `parameters`, and `fitMeasures(x)` on the `esem_fit` wrapper; `fitMeasures(x)` reads `wlsmv_stats` when present. Do not interpret `summary(x$lavaan_fit)`, `modindices(x)`, or `coef(x)` as the rotated solution unless you know the fit used the lavaan `efa()` path (`method = "lavaan"` or `method = "set-esem"` for B-ESEM).

Heywood fix and loadings

When `heywood_fix = TRUE` (default on single-group `esem` and `esem_ordered`), `std_loadings` may reflect a post-hoc rotation correction while `lavaan_fit` still holds the pre-correction lavaan solution. Prefer `std_loadings(x)` for reported loadings in that case.

References

Asparouhov, T., & Muthen, B. (2009). Exploratory structural equation modeling. *Structural Equation Modeling, 16*(3), 397–438.

See Also

`esem` for continuous data, `besem_ordered` for bifactor ordered ESEM.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

# Derive ordered (5-category Likert) versions of the 9 continuous items
items <- paste0("x", 1:9)
ord <- as.data.frame(lapply(HolzingerSwineford1939[, items], function(v) {
  as.integer(cut(v, breaks = quantile(v, probs = seq(0, 1, 0.2)),
                 include.lowest = TRUE))
}))
names(ord) <- items

tgt <- make_target(
  list(Visual = c("x1", "x2", "x3"),
       Textual = c("x4", "x5", "x6"),
```

```
      Speed = c("x7", "x8", "x9")),
  item_names = items
)


fit_ord <- esem_ordered(
  data         = ord,
  nfactors     = 3,
  indicators   = items,
  rotation     = "target",
  target       = tgt,
  factor_names = c("Visual", "Textual", "Speed"),
  n_starts     = 5L
)

summary(fit_ord, fit.measures = TRUE, standardized = TRUE)
std_loadings(fit_ord)
factor_correlations(fit_ord)
```

`ewc_syntax` *Generate ESEM-within-CFA Lavaan Syntax*

Description

Converts a fitted ESEM solution into explicit lavaan CFA syntax following the ESEM-within-CFA approach (Marsh et al. 2014). The syntax uses unstandardised ESEM loadings as starting values (`start(v)*item`) or fixed values (`v*item`) depending on the referent scheme.

Usage

```
ewc_syntax(esem_fit, spec, referents = NULL, var_fixed = TRUE)
```

Arguments

`esem_fit`	An `esem_fit` object from `run_comparison` (`results$fit_esem`) or `esem` / `esem_ordered`.
`spec`	An `esem_spec` from `specify_model`.
`referents`	Named character vector of referent items, one per factor. Names = factor names; values = item names. If `NULL` (default), referents are selected automatically via `find_ewc_referents`.
`var_fixed`	Logical. Identification mode: `TRUE` **(default)** Factor variances fixed to 1 (`std.lv = TRUE`). Only referent cross-loadings on other factors are fixed. Mirrors Mplus `EX\@1; MD\@1; CI\@1;`. `FALSE` Factor variances freely estimated. Referent's own primary loading is *also* fixed for identification. Mirrors Mplus `EX*; MD*; CI*;`. For a B-ESEM source the same two schemes apply to G and to every specific factor; all factor covariances stay fixed at zero in both.

Value

A character string of lavaan model syntax. Use `cat()` to inspect, or pass directly to `fit_ewc` via `custom_syntax`.

References

Marsh, H. W., Morin, A. J. S., Parker, P. D., & Kaur, G. (2014). Exploratory structural equation modeling. *Annual Review of Clinical Psychology, 10*, 85-110.

See Also

`find_ewc_referents`, `fit_ewc`

`extract_mplus_loadings` *Extract standardized loadings from MplusAutomation output*

Description

Convenience helper for use as the `target` of `align_loadings()`: pulls the STDYX-standardized loadings from a `MplusAutomation::readModels()` result and returns one matrix per group, in items-by-factors form with names matching common conventions.

Usage

```
extract_mplus_loadings(
  mp_out,
  standardized = "stdyx.standardized",
  items = NULL,
  factors = NULL
)
```

Arguments

`mp_out`	An object returned by `MplusAutomation::readModels()`.
`standardized`	Which Mplus section to read. Default `"stdyx.standardized"`; set to `"unstandardized"` to read raw point estimates (note: comparison with lavaan's `parameterEstimates()` is not meaningful for ESEM `efa()` blocks because lavaan reports the unrotated optimization basis).
`items`	Optional character vector. If supplied, the returned matrices use this row order; otherwise the order from Mplus is used.
`factors`	Optional character vector. If supplied, the returned matrices use this column order.

Value

A named list of matrices, one per group, each with rownames (items) and colnames (factors).

factor_correlations *Extract Factor Correlations from an esem_fit*

Description

Extracts the estimated correlation matrix among latent factors.

Usage

```
factor_correlations(x, digits = 3)
```

Arguments

x	An esem_fit object.
digits	Integer. Number of decimal places. Default 3.

Value

A symmetric matrix of factor correlations.

factor_scores *Extract Latent Factor Scores*

Description

Returns a data frame of estimated latent factor scores, one column per factor and one row per observation. Works with any esem_fit, plain lavaan object, or esem_invariance result.

Usage

```
factor_scores(
  x,
  method = c("regression", "bartlett"),
  level = "auto",
  align = NULL,
  dCFI_cutoff = -0.01,
  ...
)
```

Arguments

x	An esem_fit, lavaan S4, or esem_invariance object.
method	Character. Estimation method: "regression" (default, BLUP – minimises MSE, scores are correlated when factors are correlated) or "bartlett" (unbiased, correct factor variance). For WLSMV B-ESEM models with orthogonal rotation both methods return identical scores ($\Phi = I$).
level	Character. Only used when x is an esem_invariance object. "auto" (default) selects the most constrained fitted invariance level whose own transition ΔCFI ≥ -0.010, including "varcov" and "means" when esem_invariance() was run with through. Override with "configural", "weak", "strong", "strict", "varcov" or "means". "auto" skips a level whose reported solution is inadmissible (a negative variance or a non-positive-definite matrix, recorded in inv$notes) even when its ΔCFI passes. Silently ignored for plain fit objects.
align	Optional alignment of the score columns. NULL (default) returns scores in the model's native rotation orientation – which can sign-flip or column-permute across reruns or solvers (lavaan vs Mplus) for B-ESEM target rotation. Supply one of: • "canonical" – deterministic rule (columns sorted by $\Sigma\lambda^2$ descending, sign-flipped so the largest $\lvert\lambda\rvert$ per column is positive). No external reference; reproducible across reruns. • "group1" – multi-group only; rotate each group's scores to match group 1's loading orientation. • a numeric matrix (single-group) or list of matrices (multi-group) of reference standardized loadings (rownames = items, colnames = factors). Useful for aligning to an external reference such as Mplus output via extract_mplus_loadings(). • another fit object (esem_fit, esem_invariance) – aligns to that fit's loadings. Internally calls align_loadings() to compute the per-group rotation Q, then rotates the score matrix by Q. Note: under "canonical" columns may be permuted, in which case the column names retain their original ordering – treat them as positional after alignment. Not supported for besem_fit_ordered fits produced by besem_ordered() with custom WLSMV polychoric scoring.
dCFI_cutoff	Numeric. The ΔCFI threshold used by level = "auto" to accept an invariance level (a level qualifies when its transition ΔCFI $\geq$ dCFI_cutoff). Defaults to -0.010 (Cheung & Rensvold, 2002). Only used when x is an esem_invariance object and level = "auto".
...	Reserved for future use.

Details

For multi-group ordered B-ESEM scoring, a single pooled polychoric correlation matrix is applied to every group. A warning is issued because scores for non-reference groups may be biased when their correlation structures differ. If the polychoric correlation matrix or the score-information matrix is singular, factor_scores() raises an informative error identifying the singular matrix rather than exposing a bare solve() failure.

Value

A `data.frame` with one column per latent factor (named by the model's factor labels) and one row per observation. A `group` column is prepended automatically for multi-group models. Rows for missing observations contain `NA` in all factor columns.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]


# Single fit
fit    <- esem(d, nfactors = 3)
scores <- factor_scores(fit)
head(scores)

# Merge back to original data
d_with_scores <- cbind(d, factor_scores(fit))

# Bartlett method
scores_b <- factor_scores(fit, method = "bartlett")
```

`find_ewc_referents` *Auto-Select EWC Referent Items*

Description

For each factor, selects the primary indicator with the largest absolute unstandardised loading from the ESEM solution. This is the item that will anchor that factor's identification in the EWC model.

Usage

```
find_ewc_referents(esem_fit, spec)
```

Arguments

`esem_fit`	An `esem_fit` object (e.g. `results$fit_esem` from `run_comparison`).
`spec`	An `esem_spec` from `specify_model`.

Details

Referents are chosen solely by maximum |loading| on the primary factor. You can override automatic selection by supplying a custom named vector to the `referents` argument of `ewc_syntax` or `fit_ewc`.

Value

A named character vector: names are factor names, values are the selected referent item names (in the original case from spec).

See Also

ewc_syntax, fit_ewc

---

fit_ewc *Fit an ESEM-within-CFA Model*

Description

Generates EWC lavaan syntax from a fitted ESEM solution and estimates it as a standard lavaan::cfa() model – no rotation required.

Usage

```
fit_ewc(
  esem_fit,
  spec,
  referents = NULL,
  var_fixed = TRUE,
  missing = NULL,
  custom_syntax = NULL,
  ...
)
```

Arguments

esem_fit	An esem_fit object (results$fit_esem).
spec	An esem_spec from specify_model.
referents	Named character vector of referent items or NULL (auto-selected). See find_ewc_referents.
var_fixed	Logical. Identification mode. Default TRUE (factor variances = 1). See ewc_syntax for details.
missing	Character. Missing data handling. Defaults to "pairwise" for ordered, "listwise" for continuous.
custom_syntax	Character or NULL. Supply a hand-edited syntax string (from ewc_syntax) instead of auto-generating. When non-NULL, referents and var_fixed still control the lavaan::cfa() options (std.lv).
...	Additional arguments forwarded to lavaan::cfa().

Value

An object of class "ewc_fit":

lavaan_fit lavaan fit object; all lavaan generics work on it.

syntax lavaan model string used.

referents Named vector of referent items.

var_fixed Identification mode used.

estimator Estimator ("MLR" for continuous, "DWLS" for ordered).

spec The spec object.

See Also

ewc_syntax, find_ewc_referents, compare_ewc

---

fit_indices *Mplus-Matched Fit Indices for a Single Model*

Description

Returns CFI, TLI, RMSEA (with 90\ character vector. Uses the .scaled variants when the fit was estimated with MLR/WLSMV, matching Mplus's default output. For B-ESEM / ESEM WLSMV fits that carry pre-computed Mplus-matched values in $wlsmv_stats, those values are used directly and the RMSEA CI is computed from the scaled chi-square via non-central chi-squared inversion (MacCallum, Browne & Sugawara, 1996).

Usage

```
fit_indices(fit)
```

Arguments

fit	An esem_fit / besem_fit object, or a raw lavaan S4 fit.

Value

A named character vector with elements "CFI", "TLI", "RMSEA [90% CIs]", and "SRMR". Stack rows with rbind() (wrapped in noquote()) to build a comparison table.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]


fit_e <- esem(d, nfactors = 3)
fit_b <- besem(d, specific_factors = list(
  Visual  = c("x1", "x2", "x3"),
```

```
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9")
), n_starts = 5L)

noquote(rbind(
  ESEM  = fit_indices(fit_e),
  BESEM = fit_indices(fit_b)
))
```

`fitMeasures.esem_fit` *Extract Fit Measures from an esem_fit*

Description

Forwards to `lavaan::fitMeasures()` when the fit used lavaan's `efa()` path. When `wlsmv_stats` is present (custom DWLS/WLSMV from `besem_ordered(method = "rotation")` or `esem_ordered(method = "rotation")`), returns those indices instead.

Usage

```
## S3 method for class 'esem_fit'
fitMeasures(
  object,
 fit.measures = c("cfi", "tli", "rmsea", "rmsea.ci.lower", "rmsea.ci.upper", "srmr",
    "aic", "bic"),
  ...
)
```

Arguments

`object`	An `esem_fit` object.
`fit.measures`	Character vector of fit index names. Default returns a standard set: CFI, TLI, RMSEA, SRMR, AIC, BIC.
`...`	Passed to `lavaan::fitMeasures()`.

Value

A named numeric vector of fit indices (by default CFI, TLI, RMSEA with its confidence interval, SRMR, and, on lavaan paths, AIC and BIC).

`generate_mplus_besem_syntax` *Generate Mplus B-ESEM Syntax*

Description

Generates the correct Mplus syntax for Bifactor ESEM with orthogonal target rotation, using separate BY statements per factor with (*1).

Usage

```
generate_mplus_besem_syntax(
  specific_factors,
  all_indicators,
  g_name = "G",
  cfa_factors = NULL,
  regressions = NULL,
  covariances = NULL,
  data_file = "mydata.dat",
  missing_code = 999,
  output_path = NULL,
  ordered = FALSE,
  estimator = if (ordered) "WLSMV" else "MLR"
)
```

Arguments

`specific_factors`	Named list of specific factor -> item assignments.
`all_indicators`	Character vector of all ESEM indicator names.
`g_name`	Character. General factor name. Default `"G"`.
`cfa_factors`	Optional named list of additional CFA factors.
`regressions`	Optional character vector of regression statements.
`covariances`	Optional character vector of covariance statements.
`data_file`	Character. Data file name. Default `"mydata.dat"`.
`missing_code`	Missing value code. Default `999`. Use `"."` for files written by `MplusAutomation::prepareMplusData()` which writes `MISSING = .;`.
`output_path`	Character. Path to write .inp file. Returns syntax invisibly if `NULL`.
`ordered`	Logical. `TRUE` declares the indicators `CATEGORICAL` and keeps Mplus's default delta parameterization, the model `besem_ordered` fits: residual variances are one minus the communality, so Mplus reports a negative one (Heywood case) where the theta parameterization would not. Default `FALSE`.
`estimator`	Character. Default `"WLSMV"` when `ordered = TRUE`, otherwise `"MLR"`.

Value

Mplus syntax string (invisibly). Writes file if `output_path` supplied.

`generate_mplus_syntax` *Generate Mplus Syntax for ESEM with Target Rotation*

Description

Generates a complete Mplus .inp file for ESEM with target rotation, using the correct Mplus syntax: separate BY statements per factor, each with its own (*1) label. Primary items always appear first, followed by cross-loading items targeted to zero.

Usage

```
generate_mplus_syntax(
  factors,
  cfa_factors = NULL,
  regressions = NULL,
  covariances = NULL,
  data_file = "mydata.dat",
  missing_code = 999,
  estimator = if (ordered) "WLSMV" else "MLR",
  output_path = NULL,
  ordered = FALSE
)
```

Arguments

`factors`	Named list mapping factor names to their primary indicator names. Example: `list(EX = c("y1","y2"), MD = c("y3","y4"))`.
`cfa_factors`	Optional named list of additional CFA factors.
`regressions`	Optional character vector of regression statements.
`covariances`	Optional character vector of covariance statements.
`data_file`	Character. Name of the data file. Default `"mydata.dat"`.
`missing_code`	Missing value code. Default `999`. Use `"."` for files written by `MplusAutomation::prepareMplusData()` which writes `MISSING = .;`.
`estimator`	Character. Default `"WLSMV"` when `ordered = TRUE`, otherwise `"MLR"`.
`output_path`	Character. Full path to write the .inp file. Returns syntax string invisibly if `NULL`.
`ordered`	Logical. `TRUE` declares the factor indicators `CATEGORICAL` and uses the theta parameterization, as `esem_ordered` does. Default `FALSE`.

Details

Correct output format:

```
EX BY
  batEX1 batEX2 ... batEX8       <- primary items first
  batMD1~0 ... batCI5~0 (*1);    <- cross-loadings after

MD BY
  batMD1 batMD2 ... batMD5       <- primary items first
  batEX1~0 ... batCI5~0 (*1);    <- cross-loadings after
```

Value

The Mplus syntax as a character string (invisibly). Writes file if `output_path` is supplied.

Examples

```
# Pure syntax generation: builds a Mplus .inp string, no Mplus install needed.
syntax <- generate_mplus_syntax(
  factors = list(
    Visual  = c("x1", "x2", "x3"),
    Textual = c("x4", "x5", "x6"),
    Speed   = c("x7", "x8", "x9")
  ),
  output_path = file.path(tempdir(), "esem_model.inp")
)
cat(syntax)
```

---

get_syntax *Extract the Generated lavaan Syntax*

Description

Extract the Generated lavaan Syntax

Usage

```
get_syntax(x, cat = TRUE)
```

Arguments

`x`	An `esem_fit` object.
`cat`	Logical. If `TRUE` (default), also print to console.

Value

The lavaan model string (invisibly).

lavaan_fit *Extract the Underlying lavaan Fit Object*

Description

Returns the raw lavaan object stored in x$lavaan_fit. For continuous ESEM/B-ESEM and method = "lavaan" ordered fits, this is the fitted model. For besem_ordered(method = "rotation") (and esem_ordered(method = "rotation")), it is often an **auxiliary** one-factor CFA used for DWLS weights only—see The lavaan_fit slot on custom WLSMV fits. Use lavaan::lavTestScore(), lavaan::lavInspect(), etc. only when that slot represents the model you intend to analyse.

Usage

```
lavaan_fit(x)
```

Arguments

x	An esem_fit object.

Value

A lavaan object.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")
d <- HolzingerSwineford1939[, paste0("x", 1:9)]


fit <- esem(d, nfactors = 3)
lav <- lavaan_fit(fit)

# Inspect the underlying lavaan model
lavaan::lavInspect(lav, "cor.lv")   # factor correlations
lavaan::fitMeasures(lav, c("cfi", "rmsea"))
```

make_bifactor_target *Create a Bifactor Target Matrix*

Description

Builds the target loading matrix for B-ESEM. The general factor G has target 1 for all items (free to load everywhere). Each specific factor has target 1 for its primary items and 0 for all others.

Usage

```
make_bifactor_target(specific_factors, indicators, g_name = "G")
```

Arguments

specific_factors	Named list of specific factor -> item assignments.
indicators	Character vector of all item names.
g_name	Character. Name for the general factor. Default "G".

Value

A numeric matrix (items x factors) of class "esem_target". Column order: G first, then specific factors in list order.

Examples

```
tgt <- make_bifactor_target(
  specific_factors = list(EX = c("y1","y2","y3"),
                          MD = c("y4","y5","y6")),
  indicators       = paste0("y", 1:6)
)
print(tgt)
```

---

make_target *Create a Target Loading Matrix for Target Rotation*

Description

Constructs a target matrix suitable for use with rotation = "target" in esem. Items listed in keys are assigned a target value of 1 for their primary factor; all other cells are set to 0 (penalised to be near zero) or NA (free, no penalty).

Usage

```
make_target(keys, nitems = NULL, item_names = NULL, cross_loading_value = 0)
```

Arguments

keys	A named list where each element is a vector of item indices (integers) or item names (characters) that are hypothesised to load primarily on that factor. Names become factor names. Example: list(F1 = 1:5, F2 = 6:10, F3 = 11:15).
nitems	Integer. Total number of items. Required when keys uses integer indices and item_names is not supplied.
item_names	Optional character vector of item names of length nitems. When supplied, keys may use either names or indices. Becomes the row names of the target matrix.

`cross_loading_value`
: Numeric or `NA`. Value assigned to cells that are *not* the primary factor. Use `0` (default) to penalise cross-loadings toward zero, or `NA` to leave them completely free (soft target rotation).

Details

Target vs. Soft Target Rotation:

- **Hard target** (`cross_loading_value = 0`): Cross-loadings are penalised toward zero. Use this when you have strong theory.
- **Soft target** (`cross_loading_value = NA`): Only primary loadings are targeted; cross-loadings are completely free. Use this when you are less certain about the zero pattern.

Items Loading on Multiple Factors:

An item can appear in multiple `keys` entries if it is expected to have meaningful loadings on more than one factor. In that case its target value will be `1` for both factors and `cross_loading_value` elsewhere.

Value

A numeric matrix (items x factors) with row names set to item names and column names set to factor names from `keys`.

See Also

`esem`

Examples

```
# Simple 15-item, 3-factor target (integer keys)
tgt <- make_target(
  keys   = list(Extrav = 1:5, Agree = 6:10, Open = 11:15),
  nitems = 15
)

# Named items
tgt2 <- make_target(
  keys       = list(F1 = c("y1", "y2", "y3"), F2 = c("y4", "y5", "y6")),
  item_names = paste0("y", 1:6)
)

# Soft target (cross-loadings free)
tgt_soft <- make_target(
  keys                = list(F1 = 1:5, F2 = 6:10),
  nitems              = 10,
  cross_loading_value  = NA
)
```

modindices.esem_fit *Extract Modification Indices from an esem_fit*

Description

Forwards to `lavaan::modindices()` on `lavaan_fit(x)`. For custom WLSMV paths where `lavaan_fit` is an auxiliary one-factor CFA, indices refer to that auxiliary model, not the rotated ESEM/B-ESEM solution; see The lavaan_fit slot on custom WLSMV fits.

Usage

```
## S3 method for class 'esem_fit'
modindices(object, sort. = TRUE, maximum.number = 20, ...)
```

Arguments

`object`	An `esem_fit` object.
`sort.`	Logical. Sort by modification index value? Default `TRUE`.
`maximum.number`	Integer. Maximum number of indices to return. Default 20.
`...`	Passed to `lavaan::modindices()`.

Value

A `data.frame` of modification indices, as returned by `lavaan::modindices()`.

parameters *Display Model Parameters*

Description

Prints a formatted parameter table – standardized loadings, standard errors, z-values, and p-values – for one fit object or all three models in a pipeline. Analogous to the STDYX section of Mplus output.

Usage

```
parameters(
  x,
  model = "all",
  type = "loadings",
  suppress = 0,
  digits = 3,
  highlight_primary = TRUE
)
```

Arguments

x	An esem_fit, besem_fit, ewc_fit, or esem_comparison_pipeline object. To inspect parameters at a specific level of an esem_invariance result, pass the level's fit object directly (e.g. parameters(inv$models$strict)).
model	Character. When x is a pipeline, which models to show: "all" (default), "CFA", "ESEM", or "BESEM".
type	Character. "loadings" (default) shows only factor loadings; "all" also shows residual variances and, for ordered models, thresholds.
suppress	Numeric. Hide loadings with |z| below this value. Default 0 (show all).
digits	Integer. Decimal places. Default 3.
highlight_primary	Logical. Colour the target (primary) loadings using ANSI codes. Supported in RStudio and most terminals; falls back silently to plain output when colour is unavailable. Default TRUE.

Value

A data frame of class bifactory_parameters holding the standardized parameter table. Its columns are factor, item, std, se, z, p, and loading_type; a model column is added when x is a pipeline. With type = "all", rows for residual variances and thresholds are also included. Printing the object renders the formatted, colour-coded table; assign the result to use the values without console output.

On a Heywood-corrected fit, the rotation changed after the standard errors were computed. Consequently, se, z, and p are NA, while the corrected standardized loadings in std are retained. When suppress > 0, rows whose z is NA are dropped, as are rows below the suppression threshold.

---

parse_mplus_polychoric

Parse Mplus Polychoric Correlation Matrix from Output File

Description

Reads a Mplus .out file and assembles the polychoric (tetrachoric) correlation matrix that Mplus prints under SAMPLE STATISTICS when SAMPSTAT is requested in the OUTPUT section.

Usage

```
parse_mplus_polychoric(out_file)
```

Arguments

out_file	Character. Full path to a Mplus .out file that contains a CORRELATION MATRIX (WITH VARIANCES ON THE DIAGONAL) section. This section is written when SAMPSTAT appears in the OUTPUT block of the Mplus input file.

Details

The matrix can be passed directly to `besem_ordered` via the `r_obs_override` argument so that R's DWLS optimisation uses the same polychoric estimates as Mplus, eliminating rotational discrepancies caused by differences in the polychoric estimators.

Value

A named, symmetric numeric matrix of polychoric correlations with 1s on the diagonal. Row and column names are the item names as reported by Mplus (typically upper-case).

Examples

```
## Not run:
# Reads a polychoric matrix from a Mplus .out file (SAMPSTAT output),
# so it needs a .out produced by a licensed Mplus run.
R_mplus <- parse_mplus_polychoric(
  "path/to/besem_measurement.out"
)
dim(R_mplus)   # 18 x 18

# Use as input to besem_ordered so R rotates from the same matrix as Mplus
fit_b <- besem_ordered(
  data             = mydata,
  specific_factors = factor_items,
  r_obs_override   = R_mplus
)

## End(Not run)
```

`partial_invariance` *Partial Measurement Invariance Testing*

Description

Given a failed invariance level, uses a greedy score-test loop (`lavTestScore` + `group.partial`) to identify and free the minimum set of non-invariant equality constraints needed to restore acceptable fit (DCFI >= -0.010). Downstream invariance levels are then re-tested under the same partial constraints.

Usage

```
partial_invariance(
  inv,
  level,
  max_free = 10L,
  delta_cfi_cutoff = -0.01,
  verbose = TRUE
)
```

Arguments

inv	An esem_invariance object from esem_invariance.
level	Character. The invariance level to partially free: "strong" or "strict". Partial weak invariance is not defined for ESEM: the rotated loading block is constrained across groups as a unit, so no single loading can be freed.
max_free	Integer. Maximum parameters to free before stopping. Default 10. Byrne et al. (1989) recommend freeing the minimum number – in practice >5 rarely recovers invariance.
delta_cfi_cutoff	Numeric. Stopping criterion: loop stops when DCFI >= this value. Default -0.010 (Cheung & Rensvold, 2002).
verbose	Logical. Print progress messages. Default TRUE.

Value

An object of class "esem_partial_invariance" with:

$freed_params Data frame: round, label, group_name, score (LM), delta_cfi (baseline-relative), converged.

$partial_fit esem_fit at level with freed params.

$downstream Named list of esem_fit for levels above target.

$downstream_lrt lavTestLRT() comparisons among downstream.

$table Combined fit table (original + partial + downstream).

$converged Logical: did DCFI pass cutoff?

$group_partial Character vector: final group.partial labels.

What can be released

ESEM and B-ESEM loadings rotate as a block, so partial configural and partial weak invariance do not exist. Partial strong releases intercepts (continuous items) or thresholds (ordered items); partial strict releases residual variances. Constraints are released one at a time, largest score test first (van de Schoot, Lugtig & Hox, 2012). An intercept or threshold is released only if every factor the item belongs to keeps at least two items with invariant intercepts/thresholds (Byrne, Shavelson & Muthen, 1989), and a warning is issued when more than 20\ added at that level are released (Dimitrov, 2010). The search is data driven: report every released parameter. On ordered items the search keeps at least one threshold per item invariant (Millsap & Yun-Tein, 2004); an item's last threshold can be released by hand through esem_invariance(..., group.partial = "item|t1"), which then fixes the item's residual variance at 1 in every group. A partial refit that fits worse than the full model (a lavaan local optimum) is retried from simple starting values.

See Also

esem_invariance

plot.alignment_check *Plot Sorted Alignment Ratios*

Description

Produces a sorted alignment ratio plot (as in Figure 2 of Mehrvarz & Rouder 2026) for one or all factor pairs. Flat horizontal spread = invariant (ICM-CFA tenable). Dispersion = misspecification.

Usage

```
## S3 method for class 'alignment_check'
plot(x, pair = NULL, ...)
```

Arguments

x	An alignment_check object.
pair	Character. Name of a specific factor pair (e.g., "F1-F2"). If NULL (default), plots all pairs in a grid.
...	Ignored.

Value

Called for its side effect of drawing the alignment-ratio plot; invisibly returns NULL.

print.alignment_check *Print Method for alignment_check*

Description

Print Method for alignment_check

Usage

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

Arguments

x	An alignment_check object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the alignment-ratio summary.

print.besem_fit *Print Method for besem_fit*

Description

Print Method for besem_fit

Usage

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

Arguments

x	A besem_fit object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing a formatted B-ESEM model overview to the console.

print.bifactory_parameters *Print a Parameter Table*

Description

Renders the formatted, colour-coded parameter table produced by parameters. Called automatically when the result of parameters() is not assigned.

Usage

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

Arguments

x	A bifactory_parameters object.
...	Ignored.

Value

x, invisibly.

print.esem_comparison *Print Method for esem_comparison*

Description

Print Method for esem_comparison

Usage

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

Arguments

x	An esem_comparison object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the comparison table.

print.esem_comparison_pipeline *Print Method for esem_comparison_pipeline*

Description

Print Method for esem_comparison_pipeline

Usage

```
## S3 method for class 'esem_comparison_pipeline'
print(x, hints = TRUE, ...)
```

Arguments

x	An esem_comparison_pipeline object.
hints	Logical. Print a short help block listing the accessors available on the object (e.g. x$fit_esem). Default TRUE.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the comparison-pipeline overview.

print.esem_fit *Print Method for esem_fit*

Description

Print Method for esem_fit

Usage

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

Arguments

x	An esem_fit object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing a formatted model overview to the console.

print.esem_invariance *Print Method for esem_invariance*

Description

Print Method for esem_invariance

Usage

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

Arguments

x	An esem_invariance object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the invariance comparison table.

print.esem_partial_invariance *Print Method for esem_partial_invariance*

Description

Print Method for esem_partial_invariance

Usage

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

Arguments

x	An esem_partial_invariance object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the partial-invariance summary.

print.esem_spec *Print an esem_spec Object*

Description

Print an esem_spec Object

Usage

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

Arguments

x	An esem_spec object.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the model specification.

print.esem_target *Print a Target Rotation Matrix*

Description

Displays a `make_target()` result in a compact, readable format, marking primary loadings (`1`), penalised cells (`0`), and free cells (`NA`).

Usage

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

Arguments

`x`	An `esem_target` matrix from `make_target`.
`...`	Ignored.

Value

Invisibly returns `x`; called for the side effect of printing the target loading matrix.

print.ewc_comparison *Print Method for ewc_comparison*

Description

Displays the fit-index comparison table returned by `compare_ewc`.

Usage

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

Arguments

`x`	An `ewc_comparison` object.
`...`	Ignored.

Value

Invisibly returns `x`.

print.ewc_fit *Print Method for ewc_fit*

Description

Compact fit summary (CFI/TLI/RMSEA/SRMR and χ^2) for an ESEM-within-CFA fit.

Usage

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

Arguments

x	An ewc_fit object from fit_ewc.
...	Ignored.

Value

Invisibly returns x.

print.reliability_indices *Print a reliability_indices Object*

Description

Displays McDonald's omega and Cronbach's alpha indices following the Morin / Rodriguez et al. reporting framework.

Usage

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

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

Arguments

x	A reliability_indices object from compute_indices.
...	Ignored.

Value

Invisibly returns x; called for the side effect of printing the reliability table.

`refine_rotation` *Refine B-ESEM Rotation Using Mplus Solution as Warm Start*

Description

The bifactor target rotation criterion surface has many local minima. R and Mplus sometimes converge to different ones, leading to different partitioning of variance between G and specific factors (while fit indices remain identical because they are rotation-invariant).

Usage

```
refine_rotation(results)
```

Arguments

`results`	An `esem_comparison_pipeline` object from `run_comparison` that includes Mplus results (`results$mplus_results` must be non-NULL).

Details

This function uses orthogonal Procrustes rotation to compute the rotation matrix $\mathbf{T}$ that maps R's unrotated loading matrix toward the Mplus STDYX solution, then evaluates whether the target criterion at that $\mathbf{T}$ is lower than R's best random-start criterion. If so, the better rotation is adopted. This is methodologically sound because:

1. The same published criterion function is still being minimised.
2. Procrustes only provides a well-informed starting point; the final solution is the converged `GPArotation::targetT` optimum from that start.
3. The approach is analogous to Mansolf and Reise's (2016) recommendation to use Schmid-Leiman solutions as warm starts.

Value

The same `results` object with `fit_besem` updated if a better rotation was found, otherwise unchanged. A message reports whether the criterion improved. The `comparison_table` is also updated.

References

Mansolf, M., and Reise, S. P. (2016). Exploratory bifactor analysis: The Schmid-Leiman orthogonalization and Jennrich-Bentler analytic rotations. *Multivariate Behavioral Research*, *51*(5), 698–717.

Examples

```
## Not run:
# Requires run_comparison() results that include a Mplus rotation reference
# (results$mplus_results non-NULL), so a licensed Mplus install is needed.
results <- run_comparison(spec, mplus_folder = tempfile("mplus_"))
results <- refine_rotation(results)
omega   <- compute_omega(results)
print(omega)

## End(Not run)
```

`run_comparison` *Run the Full CFA / ESEM / B-ESEM Comparison Pipeline*

Description

Takes a model specification from `specify_model` and automatically fits CFA, ESEM, and B-ESEM in R, optionally runs all three in Mplus, and returns a comparison table plus all fitted objects.

Usage

```
run_comparison(
  spec,
  mplus_folder = NULL,
  mplus_command = "Mplus",
  run_alignment = TRUE,
  group_equal = NULL,
  n_starts = 30L
)
```

Arguments

`spec`	An `esem_spec` object from `specify_model`.
`mplus_folder`	Character. Path to a folder for Mplus files. If `NULL` (default), Mplus models are skipped.
`mplus_command`	Character. Path or command used to invoke Mplus. Default `"Mplus"` (assumes it is on PATH). Requires **MplusAutomation**.
`run_alignment`	Logical. Run alignment check? Default `TRUE`.
`group_equal`	Character vector of lavaan equality constraints (e.g. `"loadings"` for metric, `c("loadings","intercepts")` for scalar invariance).
`n_starts`	Integer. Random rotation starts for the B-ESEM rotation search. Default `30L` (matches Mplus).

Value

An object of class "esem_comparison_pipeline" containing:

spec The original model specification.

fit_cfa lavaan CFA fit object.

fit_esem esem_fit object.

fit_besem besem_fit object.

alignment alignment_check result (if run).

comparison_table Data frame of fit indices.

mplus_results List of Mplus readModels results (if run).

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data    = HolzingerSwineford1939,
  label   = "Holzinger-Swineford"
)


# Fit CFA, ESEM, and B-ESEM in R and build the comparison table
results <- run_comparison(spec, n_starts = 5L)
print(results)

# Access individual fits
summary(results$fit_esem, fit.measures = TRUE, standardized = TRUE)
std_loadings(results$fit_besem, suppress = 0.10)
factor_correlations(results$fit_esem)


## Not run:
# Also run all three models in Mplus and compare side by side.
# Requires a licensed Mplus installation reachable via `mplus_command`.
results <- run_comparison(spec, mplus_folder = tempfile("mplus_"))
print(results)

## End(Not run)
```

run_mplus_besem_invariance *Generate, Run, and Compare B-ESEM Invariance Models Against Mplus*

Description

Creates complete Mplus `.inp` files for every level fitted in `inv` (configural, weak, strong, strict and, when `esem_invariance()` was run with `through`, latent var/cov and latent means), runs them via **MplusAutomation**, then prints a side-by-side comparison of fit statistics against the R results from `esem_invariance`.

Usage

```
run_mplus_besem_invariance(
  inv,
  output_folder,
  mplus_command = "mplus",
  group_labels = NULL,
  missing_code = -999,
  difftest = TRUE
)
```

Arguments

`inv`	An `esem_invariance` object (from `model = "besem"`).
`output_folder`	Character. Folder where `data.dat`, `.inp`, and `.out` files will be written. Created if it does not exist.
`mplus_command`	Character. Full path to the Mplus executable. Default `"mplus"` (works if Mplus is on the system PATH).
`group_labels`	Named character vector mapping group values to Mplus labels, e.g. `c("1" = "MALE", "2" = "FEMALE")`. If `NULL` (default) labels are auto-derived as `G1`, `G2`, ...
`missing_code`	Numeric. Missing value sentinel written to the data file. Default `-999`.
`difftest`	Logical. Generate `DIFFTEST` constraint files for chained nested-model comparisons (strong vs weak, strict vs strong). Default `TRUE`.

Value

A data frame (invisibly) with `_R`, `_Mplus`, and `delta_` columns for each fit statistic. Printed as a table.

See Also

`esem_invariance`

Examples

```
## Not run:
# Requires a licensed Mplus installation reachable via `mplus_command`.
inv <- esem_invariance(spec, model = "besem")
cmp <- run_mplus_besem_invariance(
  inv,
  output_folder = tempfile("mplus_inv_"),
  mplus_command = "C:/Program Files/Mplus/mplus.exe",
  group_labels  = c("1" = "MALE", "2" = "FEMALE")
)

## End(Not run)
```

`save_results` *Save Pipeline Results to CSV or xlsx*

Description

Writes all standard output files to `output_folder`:

- `<label>_fit_indices.csv` – CFI/TLI/RMSEA/SRMR for all models.
- `<label>_CFA_loadings.csv` – standardised loading matrix (wide).
- `<label>_ESEM_loadings.csv` – standardised loading matrix (wide).
- `<label>_BESEM_loadings.csv` – standardised loading matrix (wide).
- `<label>_loadings_comparison.csv` – R vs Mplus loadings, long format with SEs, z-scores, p-values, and `diff_std`.
- `<label>_omega.csv` – Reliability indices (if `indices` is supplied).
- `<label>_results.xlsx` – Single xlsx workbook with all results (when `xlsx = TRUE`).

Usage

```
save_results(results, omega = NULL, output_folder, label = NULL, xlsx = FALSE)
```

Arguments

`results`	An `esem_comparison_pipeline` object from `run_comparison`.
`omega`	Optional. A `reliability_indices` object from `compute_indices`. If supplied, reliability indices are saved.
`output_folder`	Character. Path to the folder where files are written. Created if it does not exist.
`label`	Character. Prefix for output file names. Defaults to `results$spec$label`.
`xlsx`	Logical. If `TRUE`, writes a single `<label>_results.xlsx` file instead of individual CSVs. Requires the `openxlsx2` package (`install.packages("openxlsx2")`). Sheets: `Fit_Indices`, `CFA_Loadings`, `ESEM_Loadings` (if ESEM fit present), `BESEM_Loadings` (if B-ESEM fit present), `Reliability` (if `omega` supplied), `Loadings_Comparison` (if Mplus results present and both ESEM and B-ESEM fits available). Primary loadings in ESEM/BESEM sheets are bolded; G-column bolded in BESEM sheet. Default `FALSE`.

Value

Invisibly returns a character vector of file paths written.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data    = HolzingerSwineford1939,
  label   = "Holzinger-Swineford"
)


results <- run_comparison(spec, n_starts = 5L)
indices <- compute_indices(results)
save_results(results, indices,
             output_folder = file.path(tempdir(), "esem_results"))
```

---

set_target *Modify an Existing Target Matrix*

Description

Convenience function to set specific cells of a target matrix after initial construction with `make_target`.

Usage

```
set_target(target, items, factors, value)
```

Arguments

`target`	An `esem_target` matrix.
`items`	Integer indices or character names of items to modify.
`factors`	Integer indices or character names of factors to modify.
`value`	New value: `1`, `0`, or `NA`.

Value

The modified target matrix.

specify_model *Specify an ESEM Model Structure*

Description

Creates a model specification object that flows automatically through all subsequent steps – CFA, ESEM, B-ESEM, alignment check, Mplus syntax, and comparison tables. Define your factor structure once at the top; everything else is derived automatically.

Usage

```
specify_model(
  ...,
  data,
  label = "ESEM Model",
  ordered = FALSE,
  group = NULL,
  estimator = NULL,
  missing = NULL
)
```

Arguments

...	Named character vectors, one per factor. The name becomes the factor name and the vector contains the indicator names. Example: `EX = c("y1","y2","y3")`, `MD = c("y4","y5","y6")`.
data	A `data.frame` containing the indicators.
label	Optional character string labelling the model (used in output headers and Mplus titles). Default `"ESEM Model"`.
ordered	Logical or character vector. If `TRUE`, all indicators are treated as ordinal and the estimator is automatically switched to `"WLSMV"` (both R and Mplus). If a character vector of item names is supplied, only those items are treated as ordered. Default `FALSE` (continuous).
group	Character. Name of a grouping variable in `data` for multi-group models. When supplied, a configural model is fitted by default. Use `group_equal` in `run_comparison()` to test metric or scalar invariance.
estimator	Character. Override the auto-selected lavaan estimator. If `NULL` (default), uses `"WLSMV"` when `ordered` is supplied and `"MLR"` otherwise.
missing	Character. Missing-data handling. If `NULL` (default), `"pairwise"` for ordered data and `"listwise"` for continuous. Ordered accepts `"pairwise"` or `"listwise"`. Continuous accepts `"listwise"` or any FIML alias (`"fiml"`, `"ml"`, `"direct"`). The chosen method is applied uniformly to CFA, ESEM, and B-ESEM so that fit statistics are computed on the same sample (fair comparison). Note that the FIML + GPArotation::targetT path used for B-ESEM with 4+ specific factors can settle into a local optimum (G absorbs specific-factor variance); a warning is emitted when that combination is detected.

Value

An object of class "esem_spec" containing:

`factors` Named list of factor -> indicator assignments.

`factor_names` Character vector of factor names.

`all_items` Character vector of all indicators in order.

`nfactors` Number of specific factors.

`data` The supplied data frame.

`label` Model label.

`cfa_syntax` Ready-to-use lavaan CFA model string.

`target` Target matrix for ESEM target rotation.

`bifactor_target` Target matrix for B-ESEM.

Examples

```
data("HolzingerSwineford1939", package = "lavaan")

# Continuous indicators (MLR estimator by default)
spec <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data    = HolzingerSwineford1939,
  label   = "Holzinger-Swineford 3-factor"
)
print(spec)

# Multi-group spec (for invariance testing)
spec_mg <- specify_model(
  Visual  = c("x1", "x2", "x3"),
  Textual = c("x4", "x5", "x6"),
  Speed   = c("x7", "x8", "x9"),
  data    = HolzingerSwineford1939,
  group   = "sex",
  label   = "HS 3-factor (multi-group)"
)


# Run the CFA/ESEM/B-ESEM comparison from a spec
results <- run_comparison(spec)
print(results)
```

std_loadings *Extract Standardized Loadings from an esem_fit*

Description

Returns the standardized factor loading matrix. Uses `std_rotated_loadings` when present (custom WLSMV rotation path or Heywood-corrected loadings); otherwise extracts from `lavaan_fit(x)`. Analogous to Mplus STDYX loadings.

Usage

```
std_loadings(x, digits = 3, suppress = 0)
```

Arguments

`x`	An `esem_fit` object.
`digits`	Integer. Rounding digits. Default 3.
`suppress`	Numeric. Loadings with absolute value below this threshold are replaced with `NA` for readability. Default `0` (show all).

Value

A matrix of standardized loadings (items x factors).

summary.esem_fit *Summary Method for esem_fit*

Description

Prints a full summary for an `esem_fit`. For models fit through the DWLS-from-scratch path (B-ESEM ordered, or `esem_ordered(method = "rotation")`), the underlying `lavaan_fit` slot holds an auxiliary 1-factor CFA used only for weight-matrix extraction; in that case we print a self-contained bifactory summary built from the stored rotated loadings, standard errors, and WLSMV statistics. Otherwise the call is forwarded to `lavaan::summary()`.

Usage

```
## S3 method for class 'esem_fit'
summary(object, fit.measures = TRUE, standardized = TRUE, rsquare = FALSE, ...)
```

Arguments

`object`	An `esem_fit` object.
`fit.measures`	Logical. Include fit indices? Default `TRUE`.
`standardized`	Logical. Include standardized solution? Default `TRUE`.
`rsquare`	Logical. Include R-squared for endogenous variables? Default `FALSE`.
`...`	Additional arguments passed to `lavaan::summary()`.

Value

Invisibly returns `object`; called for the side effect of printing the model summary.

`summary.ewc_fit` *Summary for ESEM-within-CFA Fits*

Description

Forwards to `lavaan::lavaan-class` `summary` on the underlying lavaan fit, so you do not need to `library(lavaan)` separately.

Usage

```
## S3 method for class 'ewc_fit'
summary(
  object,
  fit.measures = TRUE,
  standardized = TRUE,
  show_loadings = TRUE,
  ...
)
```

Arguments

`object`	An `ewc_fit` object from `fit_ewc`.
`fit.measures`	Logical. Include fit indices. Default `TRUE`.
`standardized`	Logical. Include standardised estimates. Default `TRUE`.
`show_loadings`	Logical. After the lavaan summary, also print the `parameters` table with primary loadings colour-highlighted. Default `TRUE`.
`...`	Additional arguments passed to lavaan's summary method.

Value

Invisibly returns the lavaan summary object.

Index

align_loadings, 3
alignment_check, 5

besem, 8, *10–14*, *23, 24*, *29*, *32, 33*
besem_ordered, *8–11*, 12, *13, 14*, *23, 24*, *29*, *31–33*, *41, 42*, *45*, *50*
bifactory_template, 14

chisq_decomp, 15
coef.esem_fit, 16
compare_ewc, 17, *40*, *57*
compare_loadings, 18
compute_indices, 19, *20*, *58*, *63*
compute_omega, 20

esem, *7*, *10, 11*, *13*, 21, *23–25*, *29, 30*, *32–34*, *46, 47*
esem_compare, *24*, 25
esem_invariance, *9, 10*, *13*, *22, 23*, 26, *29*, *31, 32*, *51*, *62*
esem_ordered, *10, 11*, *13, 14*, *21–24*, *29*, 30, *32–34*, *41*, *43*, *45*
ewc_syntax, 34, *38–40*
extract_mplus_loadings, 35

factor_correlations, 36
factor_scores, 36
file.edit, *15*
find_ewc_referents, *34, 35*, 38, *39, 40*
fit_ewc, *17*, *35*, *38*, 39, *39*, *58*, *68*
fit_indices, 40
fitMeasures, *11*, *13, 14*, *16*, *33*
fitMeasures.esem_fit, 41

generate_mplus_besem_syntax, *11*, 42
generate_mplus_syntax, 43
get_syntax, 44

Heywood fix and loadings, *22*

lavaan_fit, 45

make_bifactor_target, *11*, 45
make_target, *22*, *24*, *31*, 46, *57*, *64*
modindices.esem_fit, 48

parameters, *11*, *14*, *16*, *33*, 48, *53*, *68*
parse_mplus_polychoric, 49
partial_invariance, 50
plot.alignment_check, 52
print.alignment_check, 52
print.besem_fit, 53
print.bifactory_parameters, 53
print.esem_comparison, 54
print.esem_comparison_pipeline, 54
print.esem_fit, 55
print.esem_invariance, 55
print.esem_partial_invariance, 56
print.esem_spec, 56
print.esem_target, 57
print.ewc_comparison, 57
print.ewc_fit, 58
print.omega_result *(*print.reliability_indices*)*, 58
print.reliability_indices, 58

refine_rotation, 59
run_comparison, *9–14*, *17–19*, *23, 24*, *29*, *32–34*, *38*, *59*, 60, *63*
run_mplus_besem_invariance, *28*, 62

save_results, 63
set_target, 64
specify_model, *9*, *11*, *13*, *24*, *26*, *29*, *33, 34*, *38, 39*, *60*, 65
std_loadings, *11*, *13, 14*, *16*, *22*, *24*, *32, 33*, 67
summary.esem_fit, 67
summary.ewc_fit, 68

The lavaan_fit slot on custom WLSMV fits, *16*, *23*, *31*, *45*, *48*