Package: umg (via r-universe)

October 22, 2026

Type Package

Title Unified Model Graphs for Statistical Models in Psychology

Version 0.6.1

Description Constructs, validates, lays out, and renders Unified Model Graphs (UMG), a formal graphical grammar for statistical models in psychology. Vertices are typed by observability, support, and inferential role; edges distinguish stochastic, symmetric, deterministic, and mixing dependence; plates encode replication and hierarchy. Diagrams can be themed and rendered with base 'grid' graphics, with 'ggplot2', as 'TikZ' code, or as 'Graphviz' DOT, and can be constructed either from one-line motif builders for many model families or automatically from 'lavaan', 'lme4', 'mirt', 'blavaan', 'brms', 'OpenMx', and 'qgraph' model objects. A diagram can be inspected as a data frame, summarised, and exported back to 'lavaan' model syntax, making the model-diagram correspondence operational in both directions. Each diagram also exposes identification diagnostics (scaling checks, a graphical counting rule, and a d-separation reader) and generates the exploratory display scaffolds implied by the model-data duality. The notation builds on established results for graphical and structural equation models: plate notation follows Buntine (1994) <doi:10.1613/jair.62>; the counting rule and scaling checks follow Bollen (1989) <doi:10.1002/9781118619179>; and the independence reader implements the d-separation criterion of Geiger, Verma and Pearl (1990) <doi:10.1002/net.3230200504> through the equivalent ancestral graph criterion of Lauritzen, Dawid, Larsen and Leimer (1990) <doi:10.1002/net.3230200503>.

License MIT + file LICENSE

URL https://hsiutingyu.github.io/umg/, https://github.com/hsiutingyu/umg

BugReports https://github.com/hsiutingyu/umg/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports grid, grDevices, stats, tools, utils

Suggests lavaan, lme4, mirt, blavaan, brms, OpenMx, qgraph, ggplot2, DiagrammeR, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Hsiu-Ting Yu [aut, cre] (ORCID: <https://orcid.org/0000-0002-3668-8033>)

Maintainer Hsiu-Ting Yu <hsiutingyu@gmail.com>

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-10-22 19:24:23 UTC

RemoteUrl https://github.com/cran/umg

RemoteRef HEAD

RemoteSha 950502de6290cf00e59d5160f57805203628f701

Contents

as.data.frame.umg . . . 3
plot.umg . . . 4
print.umg . . . 5
print.umg_identification . . . 6
summary.umg . . . 6
umg_bifactor . . . 7
umg_caterpillar . . . 8
umg_check_scaling . . . 9
umg_count_parameters . . . 10
umg_dcm . . . 11
umg_dsep . . . 12
umg_eda_scaffold . . . 13
umg_edge . . . 14
umg_esem . . . 15
umg_factor . . . 15
umg_formative . . . 16
umg_from_blavaan . . . 17
umg_from_brms . . . 18
umg_from_lavaan . . . 18
umg_from_lmer . . . 19
umg_from_mirt . . . 20
umg_from_OpenMx . . . 20
umg_from_qgraph . . . 21

umg_ggplot . . . 22
umg_growth . . . 23
umg_identify . . . 23
umg_implied_ci . . . 24
umg_irt . . . 25
umg_labelswitching . . . 26
umg_layout . . . 26
umg_lca . . . 27
umg_mediation . . . 28
umg_mimic . . . 29
umg_mixture . . . 29
umg_model . . . 30
umg_network . . . 31
umg_node . . . 32
umg_plate . . . 33
umg_render_dot . . . 34
umg_riclpm . . . 34
umg_save . . . 35
umg_secondorder . . . 36
umg_sem . . . 37
umg_theme . . . 38
umg_to_dot . . . 39
umg_to_lavaan . . . 39
umg_to_tikz . . . 40
umg_validate . . . 41

Index **42**

`as.data.frame.umg` *Coerce a UMG to a data frame*

Description

Returns the vertices or the edges of a diagram as a tidy data frame, which is convenient for inspection, tabulation in a manuscript, and programmatic edits before re-assembling with `umg_model()`.

Usage

```
## S3 method for class 'umg'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = c("edges", "vertices"),
  ...
)
```

Arguments

`x`	An object of class `umg`.
`row.names`	Unused; present for S3 consistency.
`optional`	Unused; present for S3 consistency.
`what`	`"edges"` (default) returns one row per edge with columns `from`, `to`, `kind`, `label`, and `fixed`; `"vertices"` returns one row per vertex with columns `name`, `label`, `observed`, `support`, `role`, `dist`, `fill`, and `annot`.
`...`	Ignored.

Value

A data frame.

Examples

```
as.data.frame(umg_factor("F", paste0("y", 1:3)))
as.data.frame(umg_factor("F", paste0("y", 1:3)), what = "vertices")
```

`plot.umg` *Plot a UMG with grid graphics*

Description

Renders the diagram using the same channel allocation as the TikZ lexicon: fill encodes observability, shape encodes support, border and dedicated shapes encode role, and plates are rounded rectangles. Appearance is governed by a theme (see `umg_theme()`); individual vertices may override their fill by carrying a `fill` element.

Usage

```
## S3 method for class 'umg'
plot(
  x,
  theme = umg_theme(),
  parse_labels = NULL,
  node_r = NULL,
  orientation = NULL,
  ...
)
```

Arguments

`x`	An object of class `umg` (layout computed automatically if absent).
`theme`	A `umg_theme` object (default `umg_theme()`), or a style name (`"journal"`, `"slide"`, `"cb"`).

parse_labels	Logical; parse labels as plotmath. Defaults to the theme value. TeX $ delimiters are stripped for display.
node_r	Vertex radius in native units; defaults to the theme value.
orientation	Layer direction passed to umg_layout(): "TB" (top to bottom, default) or "LR" (left to right).
...	Passed to umg_layout() when a layout is absent.

Value

Invisibly x (with layout attached).

---

print.umg *Print a Unified Model Graph*

Description

Compact console summary of a umg object: the interpretive badge, the vertex/edge/plate counts, and a tally of edge kinds.

Usage

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

Arguments

x	An object of class umg.
...	Further arguments passed to or from other methods (currently unused).

Value

The model x, invisibly.

Examples

```
print(umg_factor("F", paste0("y", 1:3)))
```

print.umg_identification

Print a UMG identification summary

Description

Formats the components collected by `umg_identify()` into a compact report: latent scaling status, the counting-rule degrees of freedom, any label-switching indeterminacy, and the number of implied conditional independencies.

Usage

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

Arguments

`x`	An object of class `umg_identification`.
`...`	Further arguments passed to or from other methods (currently unused).

Value

The object `x`, invisibly.

Examples

```
print(umg_identify(umg_factor("F", paste0("y", 1:6))))
```

summary.umg

Summarise a UMG

Description

Produces a structured summary of a diagram: vertex counts broken down by observability, support, and inferential role; edge counts by kind; the plate structure; and a one-line identification snapshot from `umg_count_parameters()` and `umg_check_scaling()`.

Usage

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

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

Arguments

`object`	An object of class `umg`.
`...`	Ignored.
`x`	An object of class `summary.umg`.

Value

An object of class `summary.umg` (a list), printed by its own method.

Examples

```
summary(umg_sem(list(F1 = paste0("y", 1:3), F2 = paste0("y", 4:6)),
                structural = list(c("F1", "F2"))))
```

---

umg_bifactor *Bifactor measurement motif*

Description

Builds a bifactor model: a general factor loading on every indicator plus one or more orthogonal group factors each loading on a contiguous block of indicators. This is the representation behind the general factor of psychopathology and similar structures.

Usage

```
umg_bifactor(indicators, groups, general = "g", plate_index = "i = 1, ..., N")
```

Arguments

`indicators`	Character vector of all indicator names.
`groups`	A list of character vectors; each element names the indicators loading on one group factor.
`general`	Name of the general factor.
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
umg_bifactor(paste0("y", 1:6),
             groups = list(g1 = paste0("y", 1:3),
                           g2 = paste0("y", 4:6)))
```

umg_caterpillar *Caterpillar (shrinkage) plot of cluster-level estimates*

Description

Renders the caterpillar plot implied by a random-coefficient vertex: cluster-level estimates ordered by value with optional uncertainty intervals, the canonical display for inspecting between-cluster variation and shrinkage. The input is the estimated deviations, which the modelling package supplies after fitting; the diagram object names the display, this function draws it.

Usage

```
umg_caterpillar(
  estimates,
  se = NULL,
  level = 0.95,
  title = "Caterpillar plot: cluster estimates"
)
```

Arguments

`estimates`	A numeric vector of cluster-level estimates, or a data frame with a column `estimate` and optionally `se` and `group`.
`se`	Optional numeric vector of standard errors (used when `estimates` is a numeric vector).
`level`	Confidence level for the interval (default `0.95`).
`title`	Plot title.

Value

A ggplot object.

Examples

```
set.seed(1)
if (requireNamespace("ggplot2", quietly = TRUE))
  umg_caterpillar(rnorm(20), se = runif(20, 0.2, 0.5))
```

umg_check_scaling *Check latent-variable scaling marks*

Description

Every latent continuous random vertex must have its scale fixed, either by a unit-valued outgoing loading (marker method), by a fixed variance (variance-standardisation method), or, for a formative composite, by a unit-valued incoming weight from an observed cause (the marker convention for composites). This function inspects each such vertex and reports whether a scaling mark is present, exposing the single most common specification error in latent variable modelling as a missing mark rather than a nonconvergence.

Usage

```
umg_check_scaling(model)
```

Arguments

model	An object of class umg.

Details

A constant parent (a const vertex with an edge into the latent vertex) fixes the *location* of the latent variable, not its scale: it pins the mean, and a variance can still be traded against the loadings. It is therefore reported separately in location_fixed and does not count as a scaling mark.

Value

A data frame with one row per latent continuous random vertex and columns vertex, scaled (logical), via (the scaling mechanism detected: "marker loading", "fixed variance", "marker weight", or "none"), and location_fixed (logical; informational, TRUE when a constant parent fixes the location).

Examples

```
umg_check_scaling(umg_factor("F", paste0("y", 1:3)))
umg_check_scaling(umg_formative(paste0("x", 1:3), outcomes = c("y1", "y2")))
```

umg_count_parameters *Graphical parameter count (t-rule)*

Description

Applies the classical counting rule for covariance-structure models: a model can be identified only if the number of free parameters does not exceed the number of distinct pieces of information the observed variables supply. The count is read from the diagram by tallying free dep edges, free cov edges, and vertex variances against the number of non-redundant observed (co)variances. The rule supplies a necessary, not sufficient, condition; a non-negative degrees of freedom does not guarantee identification.

Usage

```
umg_count_parameters(model, meanstructure = FALSE, n_means = NULL)
```

Arguments

model	An object of class umg.
meanstructure	Logical; include the observed means in the information count (p additional moments) and the free intercepts/means in the parameter count (default FALSE). See Details for how the free means are counted.
n_means	Optional integer overriding the automatic count of free mean-structure parameters when meanstructure = TRUE; ignored otherwise.

Details

With meanstructure = TRUE, the free mean-structure parameters are counted as follows. If every random vertex carries a mean annotation of the form mean = free, mean = 0, or mean = <value> in its annot field (as written by umg_from_lavaan() from a fit with a mean structure), the count is the number of vertices annotated mean = free. Otherwise the diagram does not carry intercept fixing, and the rule assumes the growth-model convention: the indicators of latent factors have intercepts fixed at zero, so the free means are the means of the latent source vertices (latent random vertices without an incoming directed edge) plus the intercepts of the observed vertices that are not regressed on a latent vertex. This reproduces lavaan's growth() count (for a linear growth model over four occasions, 14 moments, 9 free parameters, 5 degrees of freedom). For other conventions supply n_means: for a CFA fitted with free observed intercepts and zero latent means, n_means = p (the number of observed vertices), in which case the mean structure is saturated and the degrees of freedom are unchanged.

Value

A list with the data information count, the free-parameter breakdown, the total, the implied degrees of freedom, and a logical element applicable. The rule counts second-order moments and therefore applies only to models whose random vertices are all continuous; when the model contains a categorical random vertex (mixtures, latent classes, IRT, DCM), applicable is FALSE, the counts are returned as read but the degrees of freedom are set to NA, and a warning is issued.

Examples

```
umg_count_parameters(umg_factor("F", paste0("y", 1:6)))
umg_count_parameters(umg_growth(4), meanstructure = TRUE)  # df = 5
```

---

umg_dcm *Diagnostic classification model (DCM) motif*

Description

Builds a diagnostic classification (cognitive diagnosis) model: several binary latent attributes (latent categorical vertices) govern, through a Q-matrix, the observed item responses at the crossing of person and item plates. Item parameters are fixed unknowns in the item plate. This combines a latent categorical structure with the crossed-plate measurement form, a structure no single classic convention can draw.

Usage

```
umg_dcm(Q = NULL, attr_cov = TRUE, plate_index = "i = 1, ..., N")
```

Arguments

Q	A K-by-J 0/1 Q-matrix: rows are attributes, columns are items; `Q[k, j] = 1` if item `j` requires attribute `k`. Note that this is the transpose of the J-by-K (items-by-attributes) layout used by, for example, the GDINA and CDM packages; a warning is issued when the supplied matrix has at least as many rows as columns, the usual sign of the transposed convention. Because the diagram shows the generic response vertex `u_ij` rather than per-item vertices, the Q-matrix is collapsed to the attribute level: attribute `k` points at `u` when at least one item requires it. Defaults to a 3-attribute, 6-item example.
attr_cov	Logical; draw pairwise covariance edges among the latent attributes (default `TRUE`), reflecting the saturated (or higher-order) attribute distribution that standard DCMs assume. Set to `FALSE` to draw independent attributes.
plate_index	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
Q <- rbind(c(1,1,0,0,1,0), c(0,1,1,0,0,1), c(0,0,1,1,1,1))
plot(umg_dcm(Q))
```

umg_dsep *Test d-separation in a UMG*

Description

Reads a conditional-independence claim off the diagram by the d-separation criterion. Covariance edges are treated as latent common causes (the projection of an acyclic directed mixed graph onto a DAG), so that an unanalysed association behaves as a fork. The test generalises the vanishing-tetrad reasoning of the factor-analytic tradition and supplies the model's testable implications.

Usage

```
umg_dsep(model, x, y, given = character(0))
```

Arguments

`model`	An object of class `umg`.
`x`, `y`	Vertex names whose separation is queried.
`given`	Character vector of conditioning vertices (default none); may not contain `x` or `y`.

Details

The test is carried out by the moralized ancestral graph criterion (Lauritzen, Dawid, Larsen, & Leimer, 1990), which is exact and runs in time linear in the size of the graph, so dense diagrams (for example, saturated network models) are handled without approximation.

Value

`TRUE` if `x` and `y` are d-separated given `given`, `FALSE` otherwise.

Examples

```
m <- umg_mediation(direct = FALSE)
umg_dsep(m, "X", "Y")             # FALSE: connected through M
umg_dsep(m, "X", "Y", given = "M")  # TRUE: blocked by M
```

umg_eda_scaffold *Generate exploratory display scaffolds from a UMG*

Description

Implements the model-data duality of the UMG grammar (article, Table 3): each motif in the diagram induces a canonical exploratory display. Given a UMG and a data frame, the function returns a named list of ggplot objects: faceted panels for plates whose index variable is found in the data, spaghetti plots for random-coefficient motifs, scatter plots for dep edges between observed continuous vertices, latent-score distributions, mixture densities, empirical response curves, and residual correlation heat maps. The scaffolds are intentionally minimal; they are starting points for exploration, not finished graphics.

Usage

```
umg_eda_scaffold(model, data, id = NULL, time = NULL)
```

Arguments

model	An object of class umg.
data	A data frame containing the observed vertices.
id	Optional name of the cluster identifier column for spaghetti plots.
time	Optional name of the within-cluster covariate (x axis for spaghetti plots).

Value

A named list of ggplot objects (possibly empty).

Examples

```
set.seed(1)
m <- umg_factor("F", paste0("y", 1:4))
d <- as.data.frame(matrix(rnorm(400), ncol = 4,
                          dimnames = list(NULL, paste0("y", 1:4))))
umg_eda_scaffold(m, d)
```

umg_edge *Create a UMG edge*

Description

Create a UMG edge

Usage

```
umg_edge(
  from,
  to,
  kind = c("dep", "cov", "det", "mix"),
  label = "",
  fixed = NULL,
  aggregate = FALSE
)
```

Arguments

from, to	Vertex names (character scalars).
kind	Edge kind: "dep" (stochastic dependence), "cov" (symmetric covariance), "det" (deterministic assignment), or "mix" (mixture selection from a categorical vertex).
label	Optional parameter label (e.g., "$\\lambda_2$"); an edge label is shorthand for a parameter vertex (parameter promotion; see the accompanying article, Section 4.5).
fixed	Optional fixed value (e.g., 1 for a scaling loading).
aggregate	Logical; TRUE marks a deterministic (det) edge as an aggregation over a replicated index, for example a cluster mean computed from occasion-level observations. Such an edge runs from a vertex inside a plate to a vertex outside it, which the index-licensing rule W5 would otherwise refuse (see umg_validate()); marking it aggregate = TRUE exempts it from that rule. Only meaningful for det edges; default FALSE.

Value

An object of class umg_edge.

Examples

```
umg_edge("eta1", "y1", kind = "dep", fixed = 1)
umg_edge("eta1", "eta2", kind = "cov", label = "$\\psi_{21}$")
umg_edge("y", "ybar", kind = "det", aggregate = TRUE)
```

umg_esem — *Exploratory structural equation model motif*

Description

Builds an ESEM measurement block in which every factor loads on every indicator, the defining feature that distinguishes ESEM from the zero cross-loadings of confirmatory factor analysis. Factor variances are fixed to one for scaling, as rotation determines the loading pattern.

Usage

```
umg_esem(
  factors = c("F1", "F2"),
  indicators = paste0("y", 1:6),
  plate_index = "i = 1, ..., N"
)
```

Arguments

`factors`	Character vector of factor names.
`indicators`	Character vector of indicator names.
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
plot(umg_esem(c("F1", "F2"), paste0("y", 1:6)))
```

umg_factor — *Reflective factor (measurement) motif*

Description

Builds a single-factor reflective measurement model: one latent continuous factor with directed loadings onto observed continuous indicators, the first loading fixed to 1 for scaling, residual variances implied, and a person plate.

Usage

```
umg_factor(
  factor = "F",
  indicators = c("y1", "y2", "y3"),
  loadings = NULL,
  factor_label = NULL,
  indicator_labels = NULL,
  plate_index = "i = 1, ..., N",
  dist = "N(0, psi)"
)
```

Arguments

`factor`	Name of the latent factor (character scalar).
`indicators`	Character vector of indicator names (>= 2).
`loadings`	Optional character vector of loading labels for the free loadings (the indicators after the first), so of length `length(indicators) - 1`. A vector of length `length(indicators)` is also accepted, in which case its first element (the fixed marker loading) is ignored. Defaults to `lambda[k]`.
`factor_label`, `indicator_labels`	Optional display labels.
`plate_index`	Index label for the person plate.
`dist`	Distribution annotation for the factor (source vertex).

Value

An object of class `umg`.

Examples

```
m <- umg_factor("F", c("y1", "y2", "y3"))
plot(m)
```

---

umg_formative *Formative (composite) measurement motif*

Description

Builds a formative measurement model in which observed indicators are causes of a composite latent variable, the reverse of the reflective direction. Because a composite is identified only when it emits to at least two further quantities, an optional pair of reflective outcome indicators may be supplied. The scale of the composite is fixed by default by fixing the weight of the first indicator to 1 (the marker convention lavaan applies to <~ composites), so that the motif passes `umg_check_scaling()`.

Usage

```
umg_formative(
  indicators = paste0("x", 1:4),
  composite = "C",
  outcomes = NULL,
  scaling = c("marker", "none"),
  plate_index = "i = 1, ..., N"
)
```

Arguments

`indicators`	Character vector of formative indicator (cause) names.
`composite`	Name of the composite latent variable.
`outcomes`	Optional character vector of reflective outcome indicators emitted by the composite (recommended for identification, typically >= 2).
`scaling`	`"marker"` (the default) fixes the weight of the first indicator to 1; `"none"` leaves every weight free, which draws the composite without a scaling mark (it is then reported as unscaled by `umg_check_scaling()`).
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
plot(umg_formative(paste0("x", 1:4), outcomes = c("y1", "y2")))
```

---

umg_from_blavaan *Build a UMG from a blavaan model*

Description

Translates a fitted Bayesian structural equation model from the **blavaan** package into a UMG. The graph topology is identical to that of the frequentist `umg_from_lavaan()` translation; what differs is the interpretive reading, since every free parameter carries a prior. The construction makes literal the article's claim that the Bayesian and frequentist forms of a model are the same diagram differing only in which vertices have been prior-closed.

Usage

```
umg_from_blavaan(object, plate_index = "i = 1, ..., N")
```

Arguments

`object`	A fitted blavaan object.
`plate_index`	Index label for the person plate.

Value

An object of class umg.

---

umg_from_brms — *Build a UMG from a brms model*

Description

Translates a Bayesian multilevel model fitted with **brms** into a nested-plate UMG. The random-effect structure is parsed from the model formula with the same engine as umg_from_lmer(), and the fixed-effect population parameters are rendered in their prior-closed (Bayesian) form, consistent with the article's treatment of priors as parameter promotion.

Usage

```
umg_from_brms(object, data = NULL)
```

Arguments

object	A fitted brmsfit object, or a model formula using lme4-style random-effect syntax.
data	Optional data frame; only used to label index sizes.

Value

An object of class umg.

---

umg_from_lavaan — *Build a UMG from a lavaan model*

Description

Translates a lavaan parameter table into a UMG: latent variables (=~ left-hand sides) become latent continuous vertices, observed indicators become observed continuous vertices, loadings and regressions become dep edges, off-diagonal ~~ rows become cov edges, and diagonal ~~ rows are carried as variance annotations. Intercept and mean rows (~1) are carried as mean annotations on the vertex (mean = free, mean = 0, mean = <value>, or mean = fixed when lavaan fixes the mean at a sample value, as for exogenous covariates under fixed.x = TRUE), so that a model's mean structure survives the round trip through umg_to_lavaan(). A single person plate is added over all random vertices, making explicit the replication that classic SEM diagrams leave implicit.

Usage

```
umg_from_lavaan(object, plate_index = "i = 1, ..., N", bayesian = FALSE)
```

Arguments

`object`	A fitted lavaan object, or a lavaan model syntax string (which is then lavaanified without fitting).
`plate_index`	Index label for the person plate.
`bayesian`	Logical; if `TRUE`, free parameters are rendered in their prior-closed (Bayesian) form. Defaults to `FALSE`. Used by `umg_from_blavaan()`.

Value

An object of class `umg`.

`umg_from_lmer` *Build a UMG from an lme4 formula*

Description

Parses an lme4-style mixed-model formula and constructs the nested-plate UMG: the outcome and within-cluster covariates sit in the inner (occasion) plate, random coefficients are latent continuous vertices in the cluster plate, their population means are parameter vertices outside all plates (the parameter-promotion rendering of fixed vs. random effects), and random-effect covariances appear as `cov` edges.

Usage

```
umg_from_lmer(formula, data = NULL)
```

Arguments

`formula`	An lme4 model formula, e.g. `Reaction ~ Days + (Days | Subject)`.
`data`	Optional data frame; only used to label index sizes.

Value

An object of class `umg`.

umg_from_mirt *Build a UMG from a fitted mirt model*

Description

Translates a fitted item response model from the mirt package into a crossed-plate UMG: latent ability dimensions become latent continuous vertices in the person plate, the observed responses become an observed categorical vertex at the crossing of the person and item plates, and the item parameters become fixed-unknown diamonds in the item plate. The number of ability dimensions is read from the fitted object; the item-parameter glyphs are generic (discrimination and location), since the diagram represents the structure rather than per-item estimates.

Usage

```
umg_from_mirt(object, model = c("2PL", "1PL", "3PL", "graded"))
```

Arguments

`object`	A fitted `SingleGroupClass`/`mirt` object, or an integer giving the number of latent dimensions (for a quick structural sketch without a fit).
`model`	Item-model label deciding which item-parameter vertices to draw: one of `"2PL"`, `"1PL"`, `"3PL"`, `"graded"`. When `object` is a fitted mirt model and `model` is not supplied, the item type is read from the fitted object (`Rasch` maps to `"1PL"`; `graded`, `grsm`, `gpcm`, and `gpcmIRT` map to `"graded"`); an explicit `model` argument overrides the fit.

Value

An object of class `umg`.

umg_from_OpenMx *Build a UMG from an OpenMx RAM model*

Description

Translates a reticular action model (RAM) fitted or specified with **OpenMx** into a UMG. Manifest variables become observed vertices and latent variables become latent vertices; nonzero entries of the asymmetric matrix `A` become directed `dep` edges (with fixed values shown for non-free paths), and off-diagonal entries of the symmetric matrix `S` become `cov` edges. This exposes the path-diagram content of a RAM specification in the UMG lexicon.

Usage

```
umg_from_OpenMx(object, plate_index = "i = 1, ..., N")
```

Arguments

`object`	An `MxModel` containing RAM matrices `A`, `S`, and `F` (the standard `mxModel(type = "RAM")` form).
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

---

umg_from_qgraph — *Build a UMG from a qgraph object or weighted adjacency matrix*

Description

Translates a network-psychometric object into a Gaussian-graphical UMG. A **qgraph** object is reduced to its weights matrix; a plain numeric matrix is treated directly as a (partial) association matrix. Nonzero off-diagonal entries become symmetric `cov` edges among observed continuous vertices. As the article notes, the UMG displays an estimated network but does not by itself specify it, since the structure is a picture of fitted quantities.

Usage

```
umg_from_qgraph(object, threshold = 0, plate_index = "i = 1, ..., N")
```

Arguments

`object`	A `qgraph` object, or a square numeric weights matrix.
`threshold`	Absolute-value cutoff below which an edge is omitted (default `0`, i.e. keep every nonzero entry).
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

umg_ggplot *Render a UMG with ggplot2*

Description

Produces a **ggplot2** rendering of a UMG using the same visual grammar as `plot.umg()`: fill encodes observability, point shape encodes support and inferential role, plates are drawn as rectangles, and the four edge kinds are distinguished by line type and arrowhead. The returned object is an ordinary ggplot and can be further customised, faceted, or saved with `ggplot2::ggsave()`.

Usage

```
umg_ggplot(
  model,
  theme = umg_theme(),
  node_size = 14,
  text_size = 3.2,
  orientation = NULL,
  ...
)

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

Arguments

`model`	An object of class `umg`; layout is computed if absent.
`theme`	A `umg_theme` object, or a style name passed to `umg_theme()` (`"journal"`, `"slide"`, `"cb"`).
`node_size`	Point size for vertices.
`text_size`	Text size for vertex labels.
`orientation`	Layer direction passed to `umg_layout()` (`"TB"` or `"LR"`).
`...`	Passed to `umg_layout()` when a layout is absent.
`object`	An object of class `umg` (for the `autoplot` method).

Value

A ggplot object.

Examples

```
umg_ggplot(umg_factor("F", paste0("y", 1:4)))
```

umg_growth — *Latent growth-curve motif*

Description

Builds a latent growth model: a latent intercept factor and a latent slope factor over a set of manifest repeated measures, with fixed unit loadings from the intercept and fixed time-score loadings from the slope, and an intercept–slope covariance.

Usage

```
umg_growth(
  occasions = 4,
  times = NULL,
  outcome = "y",
  plate_index = "i = 1, ..., N"
)
```

Arguments

`occasions`	Number of repeated measures (>= 3).
`times`	Optional numeric time scores (defaults to 0, 1, ..., `occasions - 1`).
`outcome`	Stem for the manifest outcome names (e.g. `"y"`).
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
plot(umg_growth(4))
```

umg_identify — *Identification summary for a UMG*

Description

Collects the identification-relevant readings of a diagram into a single object: scaling-mark checks, the graphical parameter count, the label-switching flag, and the number of implied conditional independencies. Printing the object gives a compact report.

Usage

```
umg_identify(model, meanstructure = FALSE, n_means = NULL)
```

Arguments

`model`	An object of class `umg`.
`meanstructure, n_means`	Passed to `umg_count_parameters()`.

Value

An object of class `umg_identification`.

Examples

```
umg_identify(umg_factor("F", paste0("y", 1:6)))
```

---

umg_implied_ci — *Enumerate basis-set implied conditional independencies*

Description

Returns the set of conditional independencies that the model implies under the local Markov property: each vertex is independent of its non-descendants given its parents in the directed subgraph. These are the testable implications a researcher can check against the data before trusting the specification.

Usage

```
umg_implied_ci(model, observed_only = TRUE)
```

Arguments

`model`	An object of class `umg`.
`observed_only`	Logical; restrict the statements to those that are directly testable (default `TRUE`), meaning that both vertices *and every vertex in the conditioning set* are observed. Statements conditioned on latent vertices are implications of the model but cannot be checked against data directly; set `observed_only = FALSE` to list them as well.

Value

A data frame with columns `x`, `y`, and `given` (a comma-separated conditioning set), one row per implied independence. Independence is symmetric, so each statement is listed once, in canonical form: `x` precedes `y` in the model's vertex order, and the conditioning set is listed in vertex order.

Examples

```
umg_implied_ci(umg_mediation(direct = FALSE))
```

umg_irt *Item response theory motif (crossed persons x items)*

Description

Builds a unidimensional IRT diagram with crossed person and item plates: a latent ability in the person plate, item parameters as fixed unknowns in the item plate, and an observed categorical response at the crossing.

Usage

```
umg_irt(
  model = c("2PL", "1PL", "3PL", "graded", "PCM", "GPCM"),
  ability = "theta",
  n_dim = 1L
)
```

Arguments

`model`	Item model. Dichotomous: `"1PL"`, `"2PL"`, `"3PL"`. Polytomous: `"graded"` (graded response model, GRM), `"PCM"` (partial credit model), `"GPCM"` (generalized partial credit model). The choice controls which item-parameter diamonds are drawn (discrimination, difficulty/thresholds, guessing).
`ability`	Name/label stem for the latent ability.
`n_dim`	Number of latent ability dimensions (>= 1). With more than one dimension the model is multidimensional IRT (MIRT); each ability vertex points at the response.

Value

An object of class `umg`.

Examples

```
plot(umg_irt("2PL"))
plot(umg_irt("graded"))
plot(umg_irt("2PL", n_dim = 2))
```

umg_labelswitching *Flag label-switching symmetry in mixtures*

Description

Mixture and latent class models are identified only up to a permutation of the latent classes, the label-switching indeterminacy. This function reports the latent categorical random vertices whose value permutation leaves the diagram invariant, making the indeterminacy a visible property of the specification.

Usage

```
umg_labelswitching(model)
```

Arguments

model	An object of class umg.

Value

A character vector of latent categorical random vertex names (empty if the model has no mixture component).

Examples

```
umg_labelswitching(umg_mixture(umg_growth(4), c("I", "S")))
```

umg_layout *Compute a layered layout for a UMG*

Description

Assigns coordinates by longest-path layering on the directed subgraph: parameters and constants at the top, observed leaves at the bottom, vertices spread horizontally within layers. Plate rectangles are computed as padded bounding boxes of their members, with padding growing by nesting depth so nested plates remain visually distinct. Coordinates can be overridden by supplying a coords data frame, which is the recommended route for publication figures.

Usage

```
umg_layout(
  model,
  coords = NULL,
  hgap = 1.6,
  vgap = 1.8,
  orientation = c("TB", "LR")
)
```

Arguments

`model`	An object of class `umg`.
`coords`	Optional data frame with columns `name`, `x`, `y` overriding computed positions.
`hgap, vgap`	Horizontal and vertical spacing between vertices.
`orientation`	Layer direction: `"TB"` (top to bottom, the default) places sources at the top and observed leaves at the bottom; `"LR"` lays the layers out left to right. `NULL` is treated as `"TB"`.

Value

The model with a `layout` component added: a data frame of vertex coordinates and a list of plate rectangles.

---

umg_lca *Latent class motif*

Description

Builds a latent class (or latent profile) model: a latent categorical class vertex with mixing proportions, predicting a set of observed indicators. Categorical indicators yield latent class analysis; continuous indicators yield a latent profile model.

Usage

```
umg_lca(
  indicators = paste0("u", 1:4),
  categorical = TRUE,
  class = "c",
  plate_index = "i = 1, ..., N"
)
```

Arguments

`indicators`	Character vector of indicator names.
`categorical`	Logical; `TRUE` (default) for categorical indicators (LCA), `FALSE` for continuous (LPA).
`class`	Name of the latent class vertex.
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
plot(umg_lca(paste0("u", 1:4)))
```

umg_mediation *Mediation motif*

Description

Builds an X -> M -> Y mediation diagram with an optional direct effect and an optional latent confounder of the mediator and the outcome. The interpretive badge controls whether the diagram licenses a causal reading.

Usage

```
umg_mediation(
  x = "X",
  m = "M",
  y = "Y",
  direct = TRUE,
  confounder = FALSE,
  badge = c("statistical", "structural"),
  plate_index = "i = 1, ..., N"
)
```

Arguments

`x`, `m`, `y`	Names of the predictor, mediator, and outcome.
`direct`	Logical; include the direct X -> Y path (default `TRUE`).
`confounder`	Logical; include a latent confounder U of M and Y (default `FALSE`).
`badge`	`"statistical"` or `"structural"`.
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
plot(umg_mediation(confounder = TRUE, badge = "structural"))
```

umg_mimic — *MIMIC (multiple-indicator multiple-cause) motif*

Description

Builds a MIMIC model: observed covariates predict a reflective latent factor that in turn loads on observed indicators. The cause side is formative and the effect side is reflective, a structure widely used to model the effect of background variables on a latent trait and to probe differential item functioning.

Usage

```
umg_mimic(
  causes = c("z1", "z2"),
  indicators = paste0("y", 1:4),
  factor = "F",
  plate_index = "i = 1, ..., N"
)
```

Arguments

causes	Character vector of observed predictor names.
indicators	Character vector of reflective indicator names.
factor	Name of the latent factor.
plate_index	Index label for the person plate.

Value

An object of class umg.

Examples

```
plot(umg_mimic(c("age", "sex"), paste0("y", 1:4)))
```

umg_mixture — *Add a finite-mixture wrapper to an existing UMG*

Description

Promotes a set of parameter or latent vertices to class-varying quantities by adding a latent class vertex with mixing edges into them, implementing the mixture move described in the accompanying article. The latent growth model of umg_growth() wrapped this way becomes a growth mixture model.

Usage

```
umg_mixture(model, targets, class = "c", plate = NULL)
```

Arguments

model	An object of class umg.
targets	Character vector of vertex names the class selects.
class	Name of the latent class vertex to add.
plate	Optional name of an existing plate to add the class vertex to (defaults to the first plate, typically the person plate).

Value

An object of class umg.

Examples

```
g <- umg_growth(4)
umg_mixture(g, targets = c("I", "S"))
```

---

umg_model *Assemble and validate a Unified Model Graph*

Description

Assemble and validate a Unified Model Graph

Usage

```
umg_model(
  nodes,
  edges,
  plates = list(),
  badge = c("statistical", "structural"),
  validate = TRUE
)
```

Arguments

nodes	List of umg_node objects.
edges	List of umg_edge objects.
plates	List of umg_plate objects (may be empty).
badge	Interpretive mode: "statistical" (default) or "structural". Structural diagrams license causal (do-calculus) reading; statistical diagrams do not.
validate	Logical; run well-formedness checks W1-W6 (default TRUE).

Value

An object of class umg.

See Also

umg_validate(), plot.umg(), umg_to_tikz()

Examples

```
m <- umg_model(
  nodes = list(
    umg_node("eta", "$\\eta_i$", observed = FALSE, dist = "N(0, psi)"),
    umg_node("y1", "$y_{1i}$", observed = TRUE),
    umg_node("y2", "$y_{2i}$", observed = TRUE)
  ),
  edges = list(
    umg_edge("eta", "y1", "dep", fixed = 1),
    umg_edge("eta", "y2", "dep", label = "$\\lambda_2$")
  ),
  plates = list(umg_plate("person", c("eta", "y1", "y2"),
                          "i = 1, ..., N"))
)
```

---

umg_network — *Undirected network (Gaussian graphical model) motif*

Description

Builds a network-psychometric diagram: observed continuous variables joined by symmetric covariance edges representing the partial-association structure of a Gaussian graphical model.

Usage

```
umg_network(
  vars = paste0("x", 1:5),
  edges_mat = NULL,
  plate_index = "i = 1, ..., N"
)
```

Arguments

vars	Character vector of observed variable names.
edges_mat	Optional logical or numeric adjacency matrix selecting which pairs are connected; a pair is joined when either the upper- or the lower-triangular entry is nonzero, so upper-, lower-, and fully symmetric encodings are all accepted. Defaults to a fully connected graph.
plate_index	Index label for the person plate.

Value

An object of class umg.

Examples

```
plot(umg_network(paste0("x", 1:4)))
```

---

umg_node *Create a UMG vertex*

Description

Vertices are typed on three orthogonal dimensions: observability (observed vs. latent), support (continuous vs. categorical), and inferential role (random variable, fixed unknown parameter, deterministic node, or known constant).

Usage

```
umg_node(
  name,
  label = name,
  observed = FALSE,
  support = c("continuous", "categorical"),
  role = c("rv", "par", "det", "const"),
  dist = NULL,
  fill = NULL,
  annot = NULL
)
```

Arguments

name	Unique vertex identifier (character scalar).
label	Display label; defaults to name. May contain TeX math (e.g., "$\\eta_{1i}$") for TikZ export.
observed	Logical; TRUE for observed vertices. Ignored for roles "par" and "const", which are never observed data.
support	"continuous" or "categorical".
role	"rv" (random variable), "par" (fixed unknown parameter), "det" (deterministic), or "const" (known constant).
dist	Optional distribution annotation (character), e.g. "N(0, 1)". Required for source random vertices (rule W2).
fill	Optional fill colour overriding the theme default for this vertex (any R colour specification). Useful for highlighting a subset of vertices without editing the theme.
annot	Optional free-form annotation string carried with the vertex (e.g., a convergence diagnostic such as "Rhat = 1.00"). Annotations are available to renderers and exporters.

Value

An object of class umg_node.

Examples

```
umg_node("eta1", "$\\eta_{1i}$", observed = FALSE)
umg_node("c", "$c_i$", observed = FALSE, support = "categorical",
         dist = "Categorical(pi)")
umg_node("y1", "$y_{1i}$", observed = TRUE, fill = "grey60",
         annot = "reverse-scored")
```

---

umg_plate *Create a UMG plate*

Description

Plates encode exchangeable replication; nested plates encode hierarchy and overlapping plates encode crossing.

Usage

```
umg_plate(name, members, index, parent = NULL)
```

Arguments

name	Plate identifier.
members	Character vector of vertex names in the plate scope.
index	Index label, e.g. "i = 1, ..., N".
parent	Optional name of the enclosing plate (nesting).

Value

An object of class umg_plate.

Examples

```
umg_plate("person", c("eta1", "y1"), index = "i = 1, ..., N")
```

umg_render_dot — *Render a UMG through DiagrammeR (Graphviz)*

Description

Renders the DOT export of a UMG to an interactive HTML widget using **DiagrammeR**'s Graphviz engine, suitable for notebooks, Shiny, and **rmarkdown** HTML output.

Usage

```
umg_render_dot(model, ...)
```

Arguments

`model`	An object of class `umg`.
`...`	Passed to `umg_to_dot()`.

Value

A `grViz`/`htmlwidget` object.

Examples

```
umg_render_dot(umg_irt("2PL"))
```

umg_riclpm — *Random-intercept cross-lagged panel model (RI-CLPM) motif*

Description

Builds the random-intercept cross-lagged panel model of Hamaker, Kuiper, and Grasman (2015) for two constructs measured over several waves. Stable between-person differences are absorbed by two random-intercept factors, while autoregressive and cross-lagged paths operate on the within-person components, the decomposition that distinguishes the RI-CLPM from the traditional cross-lagged panel model.

Usage

```
umg_riclpm(waves = 4, x = "x", y = "y", plate_index = "i = 1, ..., N")
```

Arguments

`waves`	Number of measurement occasions (>= 2).
`x, y`	Stems for the two observed construct names.
`plate_index`	Index label for the person plate.

Value

An object of class umg.

Examples

```
plot(umg_riclpm(waves = 4))
```

---

umg_save *Save a UMG diagram to a file*

Description

Renders a UMG to a graphics file (PDF, PNG, or SVG), or writes code for one of the export backends. The output format is taken from the file extension: .pdf/.png/.svg produce graphics, .tex/.tikz produce TikZ source, and .dot/.gv produce Graphviz DOT source. Graphics may be produced with either the base grid renderer (default) or the **ggplot2** backend. Graphics dimensions are in inches.

Usage

```
umg_save(
  model,
  file,
  width = 6,
  height = 4.5,
  res = 300,
  theme = umg_theme(),
  backend = c("grid", "ggplot"),
  ...
)
```

Arguments

model	An object of class umg.
file	Output path; the extension (.pdf, .png, .svg, .tex/.tikz, .dot/.gv) selects the format.
width, height	Dimensions in inches for graphics devices.
res	Resolution in dpi for PNG output.
theme	A umg_theme for graphics output (see umg_theme()).
backend	Graphics backend for raster/vector output: "grid" (default) or "ggplot".
...	Passed to plot.umg() / umg_ggplot() (graphics), umg_to_tikz() (TikZ), or umg_to_dot() (DOT).

Value

The path file, invisibly.

Examples

```
umg_save(umg_factor("F", paste0("y", 1:4)),
         tempfile(fileext = ".pdf"))
umg_save(umg_irt("2PL"), tempfile(fileext = ".tex"))
umg_save(umg_irt("2PL"), tempfile(fileext = ".dot"))
```

---

umg_secondorder *Second-order factor motif*

Description

Builds a hierarchical factor model: a single second-order factor with directed paths to several first-order factors, each of which loads on a block of observed indicators. The second-order factor explains the covariation among the first-order factors.

Usage

```
umg_secondorder(groups, general = "g", plate_index = "i = 1, ..., N")
```

Arguments

groups	Named list mapping each first-order factor name to its indicator names.
general	Name of the second-order factor.
plate_index	Index label for the person plate.

Value

An object of class umg.

Examples

```
plot(umg_secondorder(list(F1 = paste0("y", 1:3),
                          F2 = paste0("y", 4:6),
                          F3 = paste0("y", 7:9))))
```

umg_sem *General structural equation model builder*

Description

Assembles a complete SEM from a measurement specification and an optional structural specification, the flexible route when no single-purpose motif fits. Factors named in `measurement` become latent continuous vertices; their indicators become observed continuous vertices with the first loading fixed to one for scaling (unless `scaling = "variance"`). Structural regressions and covariances are added as `dep` and `cov` edges respectively.

Usage

```
umg_sem(
  measurement,
  structural = list(),
  covariances = list(),
  scaling = c("marker", "variance"),
  observed = character(0),
  badge = c("statistical", "structural"),
  plate_index = "i = 1, ..., N"
)
```

Arguments

`measurement`	Named list mapping each latent factor name to a character vector of its indicator names.
`structural`	Optional list of length-2 character vectors `c(from, to)` giving directed regressions among factors and/or observed variables.
`covariances`	Optional list of length-2 character vectors giving symmetric covariance edges.
`scaling`	`"marker"` (fix first loading to 1, the default) or `"variance"` (fix the factor variance to 1; no marker loading). Under variance scaling, exogenous factors are annotated `N(0, 1)` and endogenous factors carry a fixed unit *residual* variance (`resid var = 1`), matching the convention of `lavaan`'s `std.lv = TRUE`, so every factor's scale is fixed.
`observed`	Optional character vector of additional observed variables referenced only in `structural`/`covariances`.
`badge`	`"statistical"` (default) or `"structural"`.
`plate_index`	Index label for the person plate.

Value

An object of class `umg`.

Examples

```
m <- umg_sem(
  measurement = list(F1 = c("y1", "y2", "y3"),
                     F2 = c("y4", "y5", "y6")),
  structural  = list(c("F1", "F2"))
)
plot(m)
```

---

umg_theme *Construct a UMG rendering theme*

Description

Returns a named list of visual constants controlling colours, sizes, and line weights for `plot.umg()`. The defaults reproduce the grayscale lexicon of the accompanying article; `style = "slide"` enlarges type and thickens strokes for projection, and `style = "journal"` is the compact grayscale default. Any individual element may be overridden through `...`.

Usage

```
umg_theme(style = c("journal", "slide", "cb"), ...)
```

Arguments

`style`	One of `"journal"` (default), `"slide"`, or `"cb"` (a colour-blind-safe palette that adds hue to the observability cue while keeping fill luminance informative).
`...`	Named overrides for any theme element, e.g. `observed_fill = "grey80"`, `label_cex = 0.9`.

Value

A named list of class `umg_theme`.

Examples

```
th <- umg_theme("slide", label_cex = 1.1)
```

umg_to_dot — *Export a UMG as Graphviz DOT code*

Description

Generates a directed-graph description in the Graphviz DOT language. Node shape encodes support and inferential role, fill encodes observability, plates become `cluster` subgraphs (nested clusters for nested plates), and the four edge kinds are rendered with distinct arrowheads and line styles. The output can be rendered by any Graphviz engine or by `umg_render_dot()`.

Usage

```
umg_to_dot(model, file = NULL, rankdir = c("TB", "LR"), theme = umg_theme())
```

Arguments

`model`	An object of class `umg`.
`file`	Optional path; when supplied, the DOT code is written there and the path returned invisibly.
`rankdir`	Graphviz layout direction: `"TB"` (default) or `"LR"`.
`theme`	A `umg_theme` controlling fills.

Value

A character vector of DOT code (invisibly when `file` is supplied).

Examples

```
cat(umg_to_dot(umg_factor("F", paste0("y", 1:3))), sep = "\n")
```

umg_to_lavaan — *Generate lavaan model syntax from a UMG*

Description

Emits **lavaan** model syntax from a diagram, the inverse of `umg_from_lavaan()`. The translation uses an explicit, documented convention. A directed (dep) edge from a latent vertex to an observed vertex is a measurement loading (=~); every other directed edge is a regression (~), so that observed-to-observed paths, observed-to-latent causes (as in MIMIC), and latent-to-latent structural paths are all written with ~. Symmetric (cov) edges become covariances (~~). Fixed edge values are written as `value * variable`; free edges are left for lavaan to estimate, and residual variances are left implicit. A vertex carrying a mean annotation in its `annot` field (as written by `umg_from_lavaan()` from a fit with a mean structure) yields an intercept line: `mean = free` becomes `v ~ 1`, and `mean = <value>` becomes `v ~ <value>*1`; `mean = fixed` (a mean fixed at a sample value, as for exogenous covariates under `fixed.x = TRUE`) emits nothing, because a refit fixes it the same way. A growth model therefore round-trips with its mean structure intact, whether refitted with `lavaan::growth()` or with `lavaan::cfa()`.

Usage

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

Arguments

`model`	An object of class `umg`.
`file`	Optional path; when supplied the syntax is written there and the path is returned invisibly.

Details

The convention cannot recover a measurement interpretation of a latent-to-latent relation (for example a second-order factor loading on first-order factors), which is written as a latent regression; a message flags this case so the relevant lines can be changed from ~ to =~ by hand if a higher-order measurement model is intended. Mixing (`mix`) and deterministic (`det`) edges, and the categorical latent structure of mixture and diagnostic models, have no basic lavaan syntax and are omitted with a warning.

Value

A single character string of lavaan syntax (invisibly when `file` is supplied).

See Also

`umg_from_lavaan()`

Examples

```
cat(umg_to_lavaan(umg_sem(
  measurement = list(F1 = paste0("y", 1:3), F2 = paste0("y", 4:6)),
  structural  = list(c("F1", "F2")))))
```

---

umg_to_tikz *Export a UMG as TikZ code*

Description

Generates a `tikzpicture` in the dialect of the umg-style.tex lexicon distributed with the package (see `system.file("tikz", "umg-style.tex", package = "umg")`), so that programmatic diagrams are stylistically identical to hand-written ones. The output compiles inside any document whose preamble loads TikZ and inputs umg-style.tex.

Usage

```
umg_to_tikz(model, file = NULL, digits = 2)
```

Arguments

`model`	An object of class `umg`; layout is computed if absent.
`file`	Optional path; when supplied the code is written there.
`digits`	Coordinate rounding (default 2).

Value

The TikZ code as a character vector, invisibly when `file` is supplied.

`umg_validate` *Validate well-formedness of a UMG*

Description

Checks the well-formedness rules of the UMG grammar: W1 acyclicity of the directed (dep/det/mix) subgraph; W2 every source vertex is a parameter, a constant, or carries an unconditional distribution annotation; W3 covariance edges join random vertices only; W4 mixing edges originate from latent categorical random vertices and never target observed vertices; W5 plates are coherent with the index structure (see Details); W6 edge endpoints refer to declared vertices. Rules are checked in linear time; violations raise errors, advisory conditions raise warnings.

Usage

```
umg_validate(model)
```

Arguments

`model`	An object of class `umg`.

Details

Rule W5 has four clauses. (a) Plate membership refers to declared vertices, parents are declared plates, and nesting is acyclic. (b) Nesting is consistent with membership: every vertex of a plate nested inside another plate also belongs to the parent plate. (c) Index licensing: for every `dep` and `det` edge u -> v, the plates of u are a subset of the plates of v, so that a quantity replicated over an index can only feed quantities that are replicated over that index too (an item parameter may feed a person-by-item response, but not a person-level ability). A `det` edge created with `aggregate = TRUE` (see `umg_edge()`) is exempt, because an aggregate such as a cluster mean legitimately collapses an index. A `cov` edge joins vertices with identical plate membership. (d) Mixing: a `mix` edge from a class vertex c selects a class-specific value of its target, so every random vertex whose density uses that value (every `dep`-child of the target) must be replicated over the plates of c. The target itself is exempt from clause (c): it denotes a K-vector from which one element is selected per copy of c, and typically sits outside the plates of c.

Value

Invisibly `TRUE` if all checks pass.

Index

as.data.frame.umg, 3
autoplot.umg *(umg_ggplot)*, 22

ggplot2::ggsave(), *22*

plot.umg, 4
plot.umg(), *22*, *31*, *35*, *38*
print.summary.umg *(summary.umg)*, 6
print.umg, 5
print.umg_identification, 6

summary.umg, 6

umg_bifactor, 7
umg_caterpillar, 8
umg_check_scaling, 9
umg_check_scaling(), *6*, *16, 17*
umg_count_parameters, 10
umg_count_parameters(), *6*, *24*
umg_dcm, 11
umg_dsep, 12
umg_eda_scaffold, 13
umg_edge, 14
umg_edge(), *41*
umg_esem, 15
umg_factor, 15
umg_formative, 16
umg_from_blavaan, 17
umg_from_blavaan(), *19*
umg_from_brms, 18
umg_from_lavaan, 18
umg_from_lavaan(), *10*, *17*, *39, 40*
umg_from_lmer, 19
umg_from_lmer(), *18*
umg_from_mirt, 20
umg_from_OpenMx, 20
umg_from_qgraph, 21
umg_ggplot, 22
umg_ggplot(), *35*
umg_growth, 23
umg_growth(), *29*
umg_identify, 23
umg_identify(), *6*
umg_implied_ci, 24
umg_irt, 25
umg_labelswitching, 26
umg_layout, 26
umg_layout(), *5*, *22*
umg_lca, 27
umg_mediation, 28
umg_mimic, 29
umg_mixture, 29
umg_model, 30
umg_model(), *3*
umg_network, 31
umg_node, 32
umg_plate, 33
umg_render_dot, 34
umg_render_dot(), *39*
umg_riclpm, 34
umg_save, 35
umg_secondorder, 36
umg_sem, 37
umg_theme, 38
umg_theme(), *4*, *22*, *35*
umg_to_dot, 39
umg_to_dot(), *34, 35*
umg_to_lavaan, 39
umg_to_lavaan(), *18*
umg_to_tikz, 40
umg_to_tikz(), *31*, *35*
umg_validate, 41
umg_validate(), *14*, *31*