Package: semnova (via r-universe)

October 26, 2026

Type Package

Title Analyzing Interindividual Differences in Experiments

Version 0.3.0.9001

Author Benedikt Langenberg [aut, cre], Axel Mayer [ctb]

Imports R6 (>= 2.0.0), grid, lavaan (>= 0.6.2), semTools, MASS, stats, utils

Suggests testthat (>= 3.0.0), ggplot2, ggraph, tidygraph, semPlot, magick, tidyr, knitr, rmarkdown

Depends R (>= 4.1.0)

Config/testthat/edition 3

Description Latent repeated measures ANOVA is a structural equation modeling based alternative to traditional repeated measures ANOVA proposed by Langenberg et al. <doi:10.1080/00273171.2020.1803038> allowing to analyse ANOVA designs with latent variables.

Maintainer Benedikt Langenberg <benedikt.langenberg@gmail.com>

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

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

Date/Publication 2026-10-26 08:02:52 UTC

RemoteUrl https://github.com/langenberg/semnova

RemoteRef HEAD

RemoteSha a3a388eb88113c758b3f9f357bb6a1abc8a77920

Contents

anova_table . . . 2
build_covariate_syntax . . . 3
build_group_weight_syntax . . . 4
build_measurement_syntax . . . 5
build_pi_layer_syntax . . . 6
build_resid_cov_syntax . . . 7
build_resid_var_syntax . . . 8
contrast_estimates . . . 8
contrast_matrix . . . 9
contrast_table . . . 10
dag . . . 10
dag_save . . . 12
expand_categorical_covariates . . . 13
fit_measures . . . 14
generate_anova_hypotheses . . . 15
hypothesis . . . 16
Hypothesis . . . 16
identification_scheme . . . 18
lgc . . . 18
Lgc . . . 20
model_parameters . . . 21
param-names . . . 22
parse_design . . . 24
plot.Lgc . . . 25
reading_raw . . . 26
regression_coefficients . . . 27
resid_set . . . 28
ResidSet . . . 29
semnova . . . 30
Semnova . . . 32
strip_construct_structure . . . 33
summary.Lgc . . . 33
test_hypothesis . . . 34
validate_hypothesis . . . 34

Index **36**

`anova_table` *Test every hypothesis attached to a fit*

Description

Runs [test_hypothesis()] over ‘fit$hypotheses‘ and stacks the omnibus tests into one table, the ANOVA table for a design fitted by [semnova()]. Hypotheses a user attached by hand are treated no differently from the ones derived from the design.

Usage

```
anova_table(fit)

## S3 method for class 'Lgc'
anova(object, ..., indices = c("rmsea", "cfi"))
```

Arguments

`fit`	An [Lgc] object.
`object`	An [Lgc] object.
`...`	Further [Lgc] objects, to compare nested models instead.
`indices`	Fit measures to report for each model, as [lavaan::fitMeasures()] names them. Each appears as a column followed by its change from the previous, less constrained model, in the order given. ‘c("rmsea", "cfi")‘ by default; ‘NULL‘ reports none. Anything ‘fitMeasures()‘ knows works, ‘c("rmsea", "cfi", "srmr")‘, say.

Value

A data.frame with one row per hypothesis, of class ‘c("anova", "data.frame")‘.

Comparing models

Given several fits, ‘anova()‘ runs a chi-square difference test through [lavaan::lavTestLRT()] and adds each model’s own fit indices, plus their change from the previous, less constrained model. The difference test alone answers only whether a constraint costs significantly, which in a large sample it usually does; invariance is conventionally judged on those descriptive changes as well.

Note that lavaan’s own ‘RMSEA‘ column is the RMSEA *of the difference test*, a different quantity from either model’s RMSEA, and is therefore shown as ‘RMSEA_D‘.

`build_covariate_syntax` *Build the covariate measurement-model + pi-regression lavaan syntax*

Description

Each latent covariate is its own construct, always measured at a single occasion (covariates are never repeated over occasions in this framework. See [n_occasions()]), so its measurement syntax is built by calling [build_measurement_syntax()] once per covariate with an occasion count of 1. That is the same [identification_scheme()] branch the etas use, reused rather than duplicated. Manifest (single-indicator) covariates need no measurement syntax at all (the "construct" is just the raw variable). Adds ‘pi ~ covariate‘ regressions for every covariate regardless of manifest/latent.

Usage

```
build_covariate_syntax(
  covariates_mmodel,
  pis,
  group_labels = "1",
  invariance_between = "strong",
  lv_scaling = c("effect", "referent"),
  n_eta_constructs = 0L
)
```

Arguments

covariates_mmodel	Named list, covariate name -> character vector of indicator names (length 1 for manifest).
pis	Character vector of pi variable names (e.g. '.pi1', '.pi2', ...).
group_labels	Character vector of group labels.
invariance_between	One of '"configural"', '"weak"', '"strong"', '"strict"'.
lv_scaling	One of '"effect"', '"referent"'.
n_eta_constructs	Integer, how many constructs the eta side already used up, so covariate labels start after them rather than colliding with them.

Value

Character(1) of lavaan model syntax ('""' if there are no covariates).

---

build_group_weight_syntax

Build the group-weight lavaan syntax

Description

By default a multigroup model treats the group sizes as fixed: they are what they are, and nothing is estimated from them. '"stochastic"' instead treats group membership as random and estimates each group's frequency, which is what you want when the grouping is itself an outcome of sampling rather than a design the study imposed. lavaan's ' weights; the model must also be fitted with 'group.w.free = TRUE'.

Each group's estimated relative frequency is added as a 'relfreq*' definition, so it appears in the parameter estimates rather than having to be worked out from the log weights by hand.

Usage

```
build_group_weight_syntax(
  group_labels,
  group_weights = c("fixed", "stochastic")
)
```

Arguments

group_labels	Character vector of group labels.
group_weights	One of ‘"fixed"‘ or ‘"stochastic"‘.

Value

Character(1) of lavaan model syntax (‘""‘ for ‘"fixed"‘, or for a single group, where there is nothing to weight).

---

build_measurement_syntax

Build the eta/indicator-layer measurement-model lavaan syntax

---

Description

Builds the CFA + invariance syntax for the latent constructs measured by ‘etas_mmodel‘. Delegates to [semTools::measEq.syntax()] except for the degenerate single-occasion, 2-indicator case, which [identification_scheme()] routes around it (see its docs) since semTools shares that gap.

Usage

```
build_measurement_syntax(
  etas_mmodel,
  group_labels = "1",
  invariance_within = "strong",
  invariance_between = "strong",
  lv_scaling = c("effect", "referent"),
  construct_offset = 0L
)
```

Arguments

etas_mmodel	Named list, construct name -> character vector of indicator names. All elements are one "occasion" of the same repeatedly-measured construct(s); ‘length(etas_mmodel)‘ is the occasion count.
group_labels	Character vector of group labels (length 1 for a single group).
invariance_within, invariance_between	One of ‘"configural"‘, ‘"weak"‘, ‘"strong"‘, ‘"strict"‘.
lv_scaling	One of ‘"effect"‘, ‘"referent"‘.

`construct_offset`
: Integer added to every construct index in the generated labels. Constructs are numbered within a call, so the covariates (each built by its own call) need an offset past the etas, or two different constructs would be given the same label and silently constrained equal. See [relabel_measurement_syntax()].

Value

Character(1) of lavaan model syntax (possibly ‘""‘ if every construct has a single indicator, i.e. nothing needs measuring), labelled by [param_label()] throughout.

`build_pi_layer_syntax` *Build the pi-layer lavaan syntax*

Description

‘pi =~ eta‘ with loadings fixed to the (inverse) within-subject contrast matrix, plus free pi intercepts and (co)variances, and eta's own (co)variance fixed to 0 (confirmed necessary via testing: ‘pi =~ eta‘ is an invertible fixed-loading transform, so any additional free variance on eta beyond what pi carries is redundant with it and leaves the vcov non-positive-definite). semTools has no equivalent. This layer (contrast-transformed latent variables) is semnova's own statistical contribution, always hand-rolled.

The covariance structure of the layer is what ‘sphericity‘, ‘compound_symmetry‘ and ‘variance_homogeneity‘ control:

* By default every pi (co)variance is free and every eta (co)variance is fixed to 0, an unstructured covariance matrix, carried entirely by the pi layer. * ‘sphericity‘ constrains a set of contrast variables to equal variances and zero covariances among themselves, the condition classical repeated-measures ANOVA assumes. Equality is imposed by giving the set one shared label, named after its lowest-numbered member. * ‘compound_symmetry‘ moves the structure to the eta layer instead: one variance shared by every occasion, one covariance shared by every pair, with the pi (co)variances fixed to 0. The shared labels use index ‘0‘, the engine's marker for "one parameter across this dimension". * ‘variance_homogeneity‘ shares those parameters across groups as well, by setting the group index to ‘0‘.

Usage

```
build_pi_layer_syntax(
  c_matrix_within,
  eta_names,
  group_labels = "1",
  sphericity = list(),
  compound_symmetry = FALSE,
  variance_homogeneity = FALSE
)
```

Arguments

`c_matrix_within`
: Numeric ‘q x q‘ contrast matrix.

`eta_names`
: Character vector naming the constructs the pi’s are measured by, one per occasion. For a latent DV these are the eta names; for a manifest (single-indicator) DV they are the manifest columns themselves, since there is no separate latent variable. Passed explicitly rather than read off the contrast matrix’s dimnames, which carry row numbers, not names.

`group_labels`
: Character vector of group labels.

`sphericity`
: List of integer vectors (or pi names), each naming a set of contrast variables to constrain to equal variances and zero covariances among themselves. Sets must not overlap. Ignored, with a warning, when ‘compound_symmetry‘ is ‘TRUE‘.

`compound_symmetry`
: Logical. Give the etas a compound-symmetric covariance matrix (one variance, one covariance) and fix the pi (co)variances to zero.

`variance_homogeneity`
: Logical. Constrain the (co)variances to be equal across groups.

Value

Character(1) of lavaan model syntax.

`build_resid_cov_syntax` *Build arbitrary residual-covariance lavaan syntax*

Description

‘measEq.syntax()‘’s ‘auto=‘ only supports blanket same-lag autocorrelation; semnova allows arbitrary user-specified indicator sets (equal or freely varying within a set), so this is always hand-rolled and called with ‘auto = 0‘ on the corresponding ‘measEq.syntax()‘ calls.

Usage

```
build_resid_cov_syntax(
  resid_cov = list(),
  equal_resid_cov = list(),
  group_labels = "1"
)
```

Arguments

`resid_cov`
: List of character vectors; residual covariances within each set are freely estimated (each pair its own parameter).

`equal_resid_cov`
: List of character vectors; residual covariances within each set are constrained equal to each other, sharing one parameter per group.

`group_labels`
: Character vector of group labels.

Value

Character(1) of lavaan model syntax (‘""‘ if both are empty).

`build_resid_var_syntax` *Build the equal-residual-variance lavaan syntax*

Description

Residual variances are freely estimated by default. Each set here is constrained instead: its members share one label, and therefore one parameter, per group. What belongs in a set is a design question – the same indicator across occasions, or one indicator within a condition , resolved by [semnova()] before it reaches here (see [resid_set()]).

Usage

```
build_resid_var_syntax(equal_resid_var = list(), group_labels = "1")
```

Arguments

`equal_resid_var`
: List of character vectors of model variables; the residual variances within each vector are constrained equal.

`group_labels`
: Character vector of group labels.

Value

Character(1) of lavaan model syntax (‘""‘ if there is nothing to constrain).

`contrast_estimates` *Estimate and test a general linear hypothesis on a fitted model*

Description

Pipeline stage 3, and the single place the contrast math happens. One function tests every effect the package reports. The default coefficient table, each ANOVA term, a contrast plot and any hypothesis you build by hand all call it with a different [Hypothesis]. There is no separate code path for "built-in" versus "custom" hypotheses.

Returns both the per-contrast estimates (with standard errors, confidence bounds, and between-person SD bands) and the omnibus Wald test of the whole hypothesis.

Usage

```
contrast_estimates(fit, hypothesis, level = 0.95)
```

Arguments

fit	An [Lgc] object.
hypothesis	A [Hypothesis] object (see [hypothesis()]).
level	Confidence level for the interval columns.

Value

An object of class ‘semnova_contrast‘: a list with ‘table‘, ‘test‘ (list with ‘stat‘, ‘df‘, ‘p.value‘), and ‘description‘.

‘table‘ is a coefficient table. It is a data.frame of class ‘"anova"‘, whose rows are labelled with the contrast names and whose columns are ‘estimate‘, ‘std_error‘, ‘z value‘, ‘ci_lower‘, ‘ci_upper‘, ‘sd_between‘ and ‘Pr(>|z|)‘. The last two names are the ones [stats::printCoefmat()] looks for, so the p-values print formatted and with significance codes; index them with backticks.

---

contrast_matrix *Kronecker contrast matrix construction*

Description

Builds the contrast matrix ‘R‘ that vectorizes a general linear hypothesis ‘H0: G . (L . B_g . M) = K‘ for a multigroup, multivariate model, per the derivation in ‘vignettes/glh.Rmd‘:

$$R = (G \otimes I_{hr})(I_{n_{groups}} \otimes (M^{\top} \otimes L))$$

applied to ‘vec(B)‘, the stacked per-group coefficient arrays.

Usage

```
contrast_matrix(G, L, M, B_array = NULL)
```

Arguments

G	Numeric matrix, ‘d x n_groups‘, the group contrast matrix.
L	Numeric matrix, ‘h x (p+1)‘, the predictor contrast matrix.
M	Numeric matrix, ‘q x r‘, the within-subject contrast matrix.
B_array	Numeric array, dimensions ‘(p+1, q, n_groups)‘, the stacked per-group coefficient matrices (predictors-incl-intercept x repeated-measures x groups).

Value

A list with ‘R‘ (the ‘(d*h*r) x (n_groups*(p+1)*q)‘ contrast matrix) and ‘dim‘ (its dimensions, named ‘c("rows", "cols")‘).

contrast_table *Per-contrast estimates for every hypothesis attached to a fit*

Description

Stacks the per-contrast tables from [contrast_estimates()] across all of ‘fit$hypotheses‘, the coefficient-level view that sits underneath [anova_table()]’s omnibus tests.

Usage

```
contrast_table(fit, level = 0.95)
```

Arguments

`fit`	An [Lgc] object.
`level`	Confidence level for the interval columns.

Value

A coefficient table: a data.frame of class ‘"anova"‘, so it prints with formatted p-values and significance codes. Rows are labelled ‘hypothesis: contrast‘, and the statistic and p-value columns are named ‘z value‘ and ‘Pr(>|z|)‘, the names [stats::printCoefmat()] recognises. Index those with backticks.

dag *Plot a model’s path diagram*

Description

Draws the fitted model as a SEM path diagram, in semnova’s column layout: residual (epsilon) nodes, indicators, the latent constructs (etas), the contrast variables (pis), then covariates and their indicators, with intercept triangles at each end.

Standard SEM conventions: single-headed arrows for directed effects (loadings, regressions, intercepts), double-headed arrows for (co)variances, self-loops for variances. Squares are manifest variables, circles latent ones, triangles the constant. Fixed parameters are marked with a ring accent over the estimate.

Multigroup fits are drawn one panel per group, each headed with the group name. Nodes stay perfectly circular, and the curved paths keep their bend, at any figure size.

Usage

```
dag(
  fit,
  what = c("measurement", "regression"),
  labels = c("all", "free", "none"),
  digits = 2,
  x_scale = 1,
  y_scale = 1,
  curve_scale = 1,
  variance_scale = curve_scale,
  covariance_scale = curve_scale,
  node_size = 9,
  text_size = 2.6,
  edge_text_size = 2.2
)
```

Arguments

`fit`	An [Lgc] object.
`what`	Which parts of the model to draw. Any of ‘"measurement"‘, ‘"regression"‘, ‘"variance"‘, ‘"covariance"‘, ‘"resid_var"‘, ‘"resid_cov"‘, ‘"intercept"‘, or ‘"all"‘ for everything. The default is the measurement and regression structure. It keeps the diagram readable, since a fully crossed design has a great many (co)variances.
`labels`	Which edges to label: ‘"all"‘ (default) labels every parameter drawn, with fixed ones shown inside a ring accent so the design is distinguishable from the estimates at a glance; ‘"free"‘ labels only freely estimated parameters, which thins out a dense diagram, the contrast loadings between the pi and eta layers are fixed, numerous, and all converge on the same nodes; ‘"none"‘ labels nothing, leaving the structure alone.
`digits`	Digits for edge labels.
`x_scale`, `y_scale`	Stretch the node positions horizontally and vertically. Only their ratio matters. They are normalised so the smaller is 1 and the larger scales up from it, since stretching both equally is the same diagram. ‘x_scale = 1.5‘ puts the columns half again as far apart while the rows keep their spacing; ‘y_scale = 1.5‘ does the same for the rows. Both ‘1‘ (the default) leaves the diagram fitted to whatever figure it is given.
`curve_scale`	How pronounced the curved (co)variance paths are. Self-loops and arcs are drawn at a fixed physical size and a fixed bend angle. ‘x_scale‘/‘y_scale‘ and the figure size leave them alone, so this is the only thing that changes them. ‘variance_scale‘ and ‘covariance_scale‘ default to it and can be set separately: the first scales the radius of the variance self-loops, the second the angle at which a covariance path leaves its node.
`variance_scale`, `covariance_scale`	Per-kind overrides of ‘curve_scale‘, which they follow unless given.
`node_size`, `text_size`, `edge_text_size`	Sizes for the node markers and the two kinds of text, in millimetres.

Details

Spacing. ‘x_scale‘ and ‘y_scale‘ stretch the node positions, and only their ratio matters: they are normalised so the tighter direction keeps its spacing at 1 and the other grows away from it, since stretching both equally is the same diagram.

The scaling fixes the *shape* of the diagram, how far apart the columns sit relative to the rows, and the whole drawing is then fitted to whatever figure it is given, shrinking uniformly rather than being cropped. The spacing is exact at ‘attr(p, "fig_size")‘, the width and height in inches the diagram was built for; below that you get a scaled-down copy, and since the markers and text keep their fixed millimetre size, a small figure crowds again. So for a dense model: scale for the shape, and give it a figure at least ‘fig_size‘ for the room.

Value

A ‘ggplot‘ object, carrying a ‘fig_size‘ attribute: the width and height in inches at which the spacing is exact.

---

dag_save *Save a path diagram, cropped to what is actually drawn*

Description

[ggplot2::ggsave()] with the whitespace taken off. A path diagram rarely fills its bounding box, the layout is fitted to the figure and the margins have to be wide enough for the labels that overhang it, so a saved file usually needs cropping by hand before it goes into a paper. This does that step: save, trim the uniform border, add back a small even margin.

Dimensions default to the plot's own ‘fig_size‘ attribute when it has one (see [dag()]), which is the size the diagram was built for.

Usage

```
dag_save(
  filename,
  plot = ggplot2::last_plot(),
  width = NULL,
  height = NULL,
  units = "in",
  dpi = 300,
  trim = TRUE,
  padding = 10,
  ...
)
```

Arguments

filename	File to write. The extension picks the format, as in ‘ggsave()‘.
plot	The plot to save, by default the last one displayed.
width, height	Figure size in ‘units‘. ‘NULL‘ (the default) takes the plot’s ‘fig_size‘ attribute if it has one, and otherwise leaves the choice to ‘ggsave()‘.
units	Units for ‘width‘ and ‘height‘.
dpi	Resolution for raster formats.
trim	Whether to crop the saved file. Requires the ‘magick‘ package and a raster format; without either, the file is still written, uncropped, with a message saying so.
padding	Margin to leave around the diagram after cropping, in pixels. ‘0‘ crops flush to the ink.
...	Passed to [ggplot2::ggsave()].

Value

The filename, invisibly.

See Also

[dag()]

Examples

```
fit <- semnova(
    data = reading_manifest, id = "id", dv = "re_fix_time",
    withins = "grade", contrasts_arg = list(grade = "contr.sum")
)
dag_save(tempfile(fileext = ".png"), dag(fit))
```

---

expand_categorical_covariates

Expand categorical covariates into contrast columns

Description

A covariate enters the model as a predictor of every contrast variable, which requires a numeric column. A categorical covariate therefore has to be coded first: ‘k‘ levels become ‘k - 1‘ contrast columns, exactly as ‘lm()‘ would code the same factor.

The columns belonging to one factor stay **one term**: they are named together in the returned mapping so that [generate_anova_hypotheses()] can test them jointly, giving a ‘k - 1‘ degree-of-freedom effect rather than ‘k - 1‘ separate one-df effects.

Usage

```
expand_categorical_covariates(
  data,
  covariates,
  contrasts_arg = list(),
  default_contrast = "contr.treatment"
)
```

Arguments

`data`	Wide-format data frame, one row per case.
`covariates`	Character vector of covariate column names.
`contrasts_arg`	Named list of contrast functions, as passed to [semnova()]. A covariate not named here uses ‘default_contrast‘.
`default_contrast`	Contrast function name for unnamed factors.

Value

A list with ‘data‘ (the frame plus any new contrast columns) and ‘terms‘ (named list, covariate -> the column names representing it; length 1 for a covariate that was already numeric).

`fit_measures` *Fit measures for a fitted model*

Description

The usual global fit indices, pulled from the underlying ‘lavaan‘ fit. Returns ‘NULL‘ for a saturated model (zero degrees of freedom), where they carry no information.

Usage

```
fit_measures(fit)
```

Arguments

`fit`	An [Lgc] object.

Value

A named numeric vector, or ‘NULL‘ if the model is saturated.

`generate_anova_hypotheses` *Generate one hypothesis per model term*

Description

Builds the ANOVA-style hypothesis set for a factorial design: one [Hypothesis] per combination of a within-subject term, a between-subject term, and a covariate term (including each design's intercept, which stands for "no term from this design"). Each hypothesis' ‘M‘/‘G‘/‘L‘ selects exactly the contrast columns/rows belonging to those terms, so testing it is testing that main effect or interaction.

The resulting hypotheses are ordinary [Hypothesis] objects, handled by [contrast_estimates()] through exactly the same path as one a user builds by hand.

Usage

```
generate_anova_hypotheses(
  within_design,
  between_design,
  n_covariates = 0L,
  covariate_names = character(),
  covariate_terms = NULL
)
```

Arguments

`within_design, between_design`	Design lists from [parse_design()].
`n_covariates`	Integer, number of covariate columns in the model.
`covariate_names`	Character vector of covariate column names.
`covariate_terms`	Named list mapping a term name to the covariate column indices that represent it, for covariates that take more than one column, a categorical covariate coded into ‘k - 1‘ contrast columns is one term with ‘k - 1‘ degrees of freedom, not ‘k - 1‘ terms. ‘NULL‘ (the default) gives one term per column.

Value

A list of [Hypothesis] objects.

`hypothesis` *Construct a general linear hypothesis*

Description

Constructor-wrapper front door for the [Hypothesis] R6 class. Prefer this over ‘Hypothesis$new()‘.

Usage

```
hypothesis(
  M = NULL,
  G = NULL,
  L = NULL,
  K = NULL,
  description = "",
  contrast_names = ""
)
```

Arguments

`M`	Numeric matrix, ‘q x r‘. The within-subject contrast matrix. May be omitted (‘NULL‘) if there is only one eta.
`G`	Numeric matrix, ‘d x n_groups‘. The group contrast matrix. May be omitted if there is only one group.
`L`	Numeric matrix, ‘h x (p+1)‘. The predictor contrast matrix. May be omitted if there are no covariates.
`K`	Numeric array, ‘d x h x r‘. The target (usually zero). May be omitted (defaults to all zeros).
`description`	Character. Human-readable hypothesis label.
`contrast_names`	Character vector. One name per row of the resulting contrast.

Value

A [Hypothesis] object.

`Hypothesis` *Hypothesis: a general linear hypothesis contrast specification*

Description

Holds the four matrices of a general linear hypothesis ‘H0: G . (L . B_g . M) = K‘ (see ‘vignettes/glh.Rmd‘). A pure data container, all field validation/defaulting logic lives in [validate_hypothesis()], not in this class; construct with [hypothesis()] rather than ‘Hypothesis$new()‘ directly.

Public fields

`M` Numeric matrix, ‘q x r‘. The within-subject contrast matrix.

`G` Numeric matrix, ‘d x n_groups‘. The group contrast matrix.

`L` Numeric matrix, ‘h x (p+1)‘. The predictor contrast matrix.

`K` Numeric array, ‘d x h x r‘. The target (usually zero).

`description` Character. Human-readable hypothesis label.

`contrast_names` Character vector. One name per row of the resulting contrast.

Methods

Public methods:

- `Hypothesis$new()`
- `Hypothesis$clone()`

Method `new()`**:** Construct a Hypothesis. Prefer [hypothesis()].

Usage:

```
Hypothesis$new(
  M = NULL,
  G = NULL,
  L = NULL,
  K = NULL,
  description = "",
  contrast_names = ""
)
```

Arguments:

`M`, `G`, `L`, `K` See field docs. May be ‘NULL‘ and filled in later by [validate_hypothesis()] when omittable (single eta/group/no covariates).

`description`, `contrast_names` See field docs.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Hypothesis$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

identification_scheme *Identification scheme for a latent construct's measurement model*

Description

Decides how many identification constraints a latent construct needs, based on how many occasions it's measured at and how many indicators it has, **not** indicator count alone.

A construct measured at a single occasion with exactly 2 indicators is under-identified by referent-indicator or effects-coding partial fixing (confirmed empirically: ‘semTools::measEq.syntax()‘ shares this gap, 2026-09-03), it needs full tau-equivalence (both loadings fixed to 1, both intercepts fixed to 0) regardless of the requested ‘lv_scaling‘. A construct measured at more than one occasion (or with a single indicator, or with 3+ indicators at a single occasion) is fine with the standard scheme.

Usage

```
identification_scheme(
  n_occasions,
  n_indicators,
  lv_scaling = c("effect", "referent")
)
```

Arguments

n_occasions	Integer(1). How many occasions this construct is measured at (1 for any covariate, always).
n_indicators	Integer(1). How many indicators this construct has.
lv_scaling	One of ‘"effect"‘, ‘"referent"‘. Ignored when the degenerate case applies.

Value

A list with ‘method‘, one of ‘"trivial"‘ (single indicator: no measurement model needed at all, the construct *is* its indicator), ‘"full_tau_equivalence"‘ (the degenerate case above), ‘"referent"‘, or ‘"effect_coding"‘.

lgc *Fit a latent growth components model*

Description

The low-level entry point: you supply the latent constructs (‘etas‘) and the within-subject contrast matrix directly, in the model's own vocabulary. For an ANOVA-style interface that derives those from a factorial design, use [semnova()].

Usage

```
lgc(
  data,
  etas,
  c_matrix_within,
  group = NULL,
  group_labels = NULL,
  covariates = NULL,
  resid_cov = list(),
  equal_resid_cov = list(),
  equal_resid_var = list(),
  invariance_within = "strong",
  invariance_between = "strong",
  lv_scaling = c("effect", "referent"),
  sphericity = FALSE,
  compound_symmetry = FALSE,
  variance_homogeneity = FALSE,
  group_weights = c("fixed", "stochastic"),
  hypotheses = NULL,
  ...
)
```

Arguments

data	A data.frame in wide format: one row per case, one column per indicator x occasion.
etas	Named list, eta name -> character vector of indicator names, one element per measurement occasion; or a bare character vector for a single eta. Every eta must have the same number of indicators. Single-indicator (manifest) etas default to being named after their indicator.
c_matrix_within	Numeric square contrast matrix, one row/column per occasion.
group	Character(1) or NULL. Column in ‘data‘ identifying groups.
group_labels	Character vector of group labels; defaults to the sorted unique values of ‘data[[group]]‘.
covariates	Named list (covariate name -> indicator name(s)) or a character vector of manifest covariate columns.
resid_cov	List of character vectors of model variables; residual covariances within each set are freely estimated. (Documented separately from ‘equal_resid_cov‘ so that [semnova()], which takes [resid_set()] objects for the same job and has no ‘equal_resid_cov‘, can inherit one without the other.)
equal_resid_cov	List of character vectors of model variables; residual covariances within each set are constrained equal to each other.
equal_resid_var	List of character vectors of model variables whose residual *variances* are constrained equal within each set. Sets may not overlap, and are ignored with a

warning under ‘"strict"‘ invariance, which already equates those parameters. Use [semnova()]’s ‘resid_var‘ to describe these by indicator and condition instead of by variable.

`invariance_within`, `invariance_between`
: One of ‘"configural"‘, ‘"weak"‘, ‘"strong"‘, ‘"strict"‘.

`lv_scaling`
: One of ‘"effect"‘ (average loading 1, average intercept 0) or ‘"referent"‘ (first loading 1, first intercept 0). Ignored where [identification_scheme()] requires full tau-equivalence.

`sphericity`
: List of index vectors (or contrast-variable names). Each set is constrained to equal variances and zero covariances among its members, the covariance structure classical repeated-measures ANOVA assumes. Sets must not overlap. ‘TRUE‘ puts every contrast variable in one set; ‘FALSE‘ (the default) leaves the covariance matrix unstructured. [semnova()] additionally accepts model terms, which it translates to positions.

`compound_symmetry`
: Logical. Structure the eta layer instead: one variance for every occasion and one covariance for every pair, with the contrast variables’ own (co)variances fixed to zero. Takes precedence over ‘sphericity‘, which is then ignored with a warning.

`variance_homogeneity`
: Logical. Constrain those (co)variances to be equal across groups.

`group_weights`
: One of ‘"fixed"‘ (default, group sizes are what they are) or ‘"stochastic"‘ (group membership is random, so each group’s frequency is estimated and reported as a ‘relfreq*‘ definition).

`hypotheses`
: A [Hypothesis] object or list of them, to attach to the fit.

`...`
: Passed to [lavaan::sem()].

Value

An [Lgc] object.

`Lgc` *Lgc: a fitted latent growth components model*

Description

Pure data container for a fitted model: fields only, no behaviour. Everything you do with a fit is a plain function taking it as an argument: [model_parameters()], [regression_coefficients()], [contrast_estimates()], [test_hypothesis()]. Construct with [lgc()].

Public fields

`sem_obj` The fitted ‘lavaan‘ object.

`syntax` Character(1). The generated lavaan syntax.

`etas_mmodel` Named list, eta name -> indicator names.

`pis` Character vector of pi (contrast) variable names.

`c_matrix_within` The within-subject contrast matrix.

`covariates_mmodel` Named list, covariate name -> indicator name(s).

`group` Character(1) or NULL. Grouping column in the data.

`group_labels` Character vector of group labels.

`hypotheses` List of [Hypothesis] objects.

`data` The data the model was fit to.

Methods

Public methods:

- `Lgc$new()`
- `Lgc$clone()`

Method `new()`**:** Construct an Lgc container. Prefer [lgc()].

Usage:

```
Lgc$new(...)
```

Arguments:

`...` Fields to set, by name.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Lgc$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`model_parameters` *Read a fitted model's parameters*

Description

Pipeline stage 1: the one canonical read of the fitted 'lavaan' object. Everything that needs to know what's in a fitted model, printing, plotting, and [regression_coefficients()] (stage 2), goes through this, so there is exactly one place that turns a 'lavaan' fit into semnova's own view of it.

'type' is classified structurally, from the operator and whether the variables involved are latent, rather than by parsing labels, a fixed parameter carries no label at all, so structure is the only thing every row has.

Every label in the model is now the engine's, including the measurement layer's (see [relabel_measurement_syntax()]), so each row also carries its parsed identity: 'semnova_type' and the index columns. An index of '0' means the parameter is one shared parameter across that whole dimension, the equality constraint an invariance level imposes.

Usage

```
model_parameters(fit, what = "all")
```

Arguments

fit	An [Lgc] object.
what	Character vector of ‘type‘ values to keep, or ‘"all"‘.

Value

A data.frame with ‘lhs‘, ‘op‘, ‘rhs‘, ‘group‘, ‘group_index‘, ‘label‘, ‘est‘, ‘se‘, ‘pvalue‘, ‘free‘, ‘type‘, ‘semnova_type‘, and the parsed index columns ‘construct‘, ‘construct2‘, ‘indicator‘, ‘indicator2‘, ‘pi‘, ‘pi2‘, ‘covariate‘ (‘NA‘ where the row has no label, ‘0‘ where the parameter is shared across that dimension).

---

param-names *Parameter naming engine*

Description

semnova identifies model parameters (loadings, intercepts, residual (co)variances, pi-layer regression coefficients, ...) by fixed-format lavaan labels. ‘param_label()‘ builds one from a parameter’s identity when writing lavaan syntax, in any of three renderings. ‘parse_param_label()‘ reads a label’s type and indices back off a fitted model. ‘regression_labels()‘ enumerates the one family a caller needs as a positioned array rather than one label at a time.

Every label the package uses is produced or parsed here. Nothing else should ever ‘paste0()‘ a parameter label.

Deliberately not an object: every piece of information a label depends on is already available wherever a label is needed, either as a function parameter during model construction or as a field on a fitted ‘Lgc‘/ ‘Semnova‘ object afterwards.

Labels encode by *index*, not by name (‘.lambda_3_2_1‘, not ‘.lambda_grade1_sentence_Y2_group1‘), so that construct/indicator/group names (which may themselves contain underscores) never have to be embedded in, and re-extracted from, the label string.

Usage

```
param_label(
  type,
  construct = NULL,
  construct2 = NULL,
  indicator = NULL,
  indicator2 = NULL,
  pi = NULL,
  pi2 = NULL,
  covariate = NULL,
  group = NULL,
```

```
  set = NULL,
  format = c("syntax", "text", "plotmath")
)

regression_labels(pis, covariates = character(), groups = NULL)

parse_param_label(label)

pretty_label(
  label,
  group_labels = NULL,
  construct_labels = NULL,
  format = c("text", "plotmath", "syntax")
)

n_occasions(construct, n_within_occasions, covariate_names = character())
```

Arguments

`type`	Character. One of ‘"loading"‘, ‘"intercept"‘, ‘"resid_var"‘, ‘"resid_cov"‘, ‘"resid_var_set"‘, ‘"resid_cov_set"‘, ‘"latent_mean"‘, ‘"latent_var"‘, ‘"latent_cov"‘, ‘"pi_intercept"‘, ‘"pi_var"‘, ‘"pi_cov"‘, ‘"regression"‘, ‘"group_weight"‘.
`construct`, `construct2`	Integer index of a construct (eta or covariate).
`indicator`, `indicator2`	Integer index of an indicator within a construct.
`pi`, `pi2`	Integer index of a pi (contrast) variable.
`covariate`	Integer index of a covariate.
`group`	Integer index of a group.
`set`	Integer index of a constrained set of residual terms (see [resid_set()]), whose members share one parameter.
`format`	How to render the parameter. ‘"syntax"‘ (the default) is the lavaan label the model is written with; ‘"text"‘ and ‘"plotmath"‘ are the readable forms for printing and for plot annotations. The identity of a parameter is its type and indices, so all three come from the same arguments. No caller should have to build a label string and hand it back to be re-parsed.
`pis`	Character or integer vector naming/indexing the pi variables.
`covariates`	Character vector of covariate names (used only for ‘dimnames‘; indices are derived from position).
`groups`	Character vector of group labels (used only for ‘dimnames‘; indices are derived from position).
`label`	Character(1). A label produced by [param_label()].
`group_labels`	Character vector mapping a group index back to its name.
`construct_labels`	Character vector mapping a construct index back to its name.

`n_within_occasions`
: Integer, total occasions in the within-design (e.g. ‘length(etas_mmodel)‘).

`covariate_names`
: Character vector of covariate names.

Functions

- `param_label()`: One label for one fully-specified parameter.
- `regression_labels()`: The label array for the pi-layer regression coefficients: ‘(covariate x pi x group)‘, with ‘"(Intercept)"‘ as the first covariate row. This is the one family that has to be enumerated rather than built one label at a time, because [regression_coefficients()] fills an array of estimates from it and the two must agree position for position. That array is the ‘B‘ of the general linear hypothesis (see ‘vignette("glh")‘). It is stated on the pi layer, so it holds no loadings, intercepts or residuals. Everything else is named with [param_label()] at the point of use.
- `parse_param_label()`: Inverse of [param_label()]: recover a label’s type and indices. The label format is fixed, so parsing needs no design data. Returns ‘list(type = NA_character_)‘ if ‘label‘ was not produced by [param_label()] (e.g. it’s a raw manifest variable name).
- `pretty_label()`: Formatting for a label you already have as a string, such as one read off a fitted model. Prefer ‘param_label(format =)‘ wherever the parameter’s identity is known – this exists because the measurement layer is generated by ‘semTools::measEq.syntax()‘ and carries labels in semTools’ own format, which can only be read back from the string. semnova’s own labels are routed through [parse_param_label()] and formatted from their identity, so the two agree. ‘format = "syntax"‘ is the identity function; ‘"plotmath"‘ gives ‘"""‘ rather than the raw label for anything that is not a parameter, since a caller about to ‘parse()‘ it has no use for a variable name. Vectorised over ‘label‘.
- `n_occasions()`: Occasion count for a named construct: ‘1‘ for any covariate (covariates are never repeated over occasions in this framework, a direct corollary of the identification-scheme guard, see ‘../decisions.md‘); the shared within-design occasion count for any eta.

`parse_design` *Parse a factorial design into contrast matrices*

Description

Turns a set of factor columns into the design objects the model needs: the crossed level grid, the design formula, the model matrix (‘b_matrix‘) built from R’s own contrast machinery, and its inverse (‘c_matrix‘). Used for the within-subject design, and, with different contrast defaults, the between-subject design.

Usage

```
parse_design(
  data,
  factors,
  contrasts_arg = list(),
```

```
  default_contrast = "contr.poly",
  normalize_contrasts = FALSE
)
```

Arguments

data	A data.frame.
factors	Character vector of column names, or 'NULL' for a design with no factors (a single implicit cell).
contrasts_arg	Named list, factor name -> contrast function name (e.g. '"contr.sum"').
default_contrast	Contrast used for any factor not named in 'contrasts_arg'.
normalize_contrasts	Logical; scale each contrast column to unit length.

Value

A list with 'idata', 'idesign', 'b_matrix', 'c_matrix', 'effect_names', 'effect_indices', and 'levels'.

---

plot.Lgc *Plot the estimated contrasts of a hypothesis*

Description

Visualises one hypothesis' contrast estimates. Built directly on [contrast_estimates()] (the same call [test_hypothesis()] makes) so the plot and the test can never disagree about what was estimated.

Give it either a 'hypothesis' object or the 'terms' of a model term (e.g. 'c("grade", "sentence")' for the grade-by-sentence interaction), which is matched against the hypotheses derived from the design.

Usage

```
## S3 method for class 'Lgc'
plot(
  x,
  hypothesis = NULL,
  terms = NULL,
  geom = c("point", "line"),
  interval = c("ci", "sd", "none"),
  level = 0.95,
  ...
)
```

Arguments

x	An [Lgc] object.
hypothesis	A [Hypothesis] object. Optional if ‘terms‘ is given.
terms	Character vector naming a model term, matched against the fit’s hypotheses. Optional if ‘hypothesis‘ is given.
geom	One or more of ‘"point"‘, ‘"line"‘, ‘"bar"‘.
interval	‘"ci"‘ for confidence intervals, ‘"sd"‘ for between-person SD bands, ‘"none"‘ for no error bars.
level	Confidence level, when ‘interval = "ci"‘.
...	Unused.

Value

A ‘ggplot‘ object.

---

reading_raw *Example data from reading research.*

Description

Example data from reading research.

Usage

```
reading_raw

reading_latent

reading_manifest
```

Format

id Unique identifier.

dv Continuous dependent variable. The variable is on its raw scale for `reading_raw`. The variable is log-transformed and standardized (grouped by indicator) for `reading_latent`.

indicator Includes the names of three indicators measuring reading skills (also see gaze_dur, tv_dur and ini_fix_dur).

sentence The type of the sentence: sentence vs. landolt.

grade Grade variable: grade0 to grade5.

gaze_dur Mean gaze duration per participant.

tv_dur Mean total viewing duration per participant

ini_fix_dur Mean initial fixation duration per participant.

age_t0 Age at grade 0.

sex Sex.

grade_t0 Grade at grade 0. Important if participant had to repeat a year.

iq IQ.

cbcl_ges_tw_t0 Child Behavior Checklist. Overall t-score at grade 0.

cbcl_int_verh_tw_t0 Child Behavior Checklist. Internalizing behavior t-score at grade 0.

cbcl_ext_verh_tw_t0 Child Behavior Checklist. Externalizing behavior t-score at grade 0.

cbcl_ges_rw_t0 Child Behavior Checklist. Overall raw score at grade 0.

cbcl_int_verh_rw_t0 Child Behavior Checklist. Internalizing behavior raw score at grade 0.

cbcl_ext_verh_rw_t0 Child Behavior Checklist. Externalizing behavior raw score at grade 0.

cbcl_ges_tw_t5 Child Behavior Checklist. Overall t-score at grade 5.

cbcl_int_verh_tw_t5 Child Behavior Checklist. Internalizing behavior t-score at grade 5.

cbcl_ext_verh_tw_t5 Child Behavior Checklist. Externalizing behavior t-score at grade 5.

cbcl_ges_rw_t5 Child Behavior Checklist. Overall raw score at grade 5.

cbcl_int_verh_rw_t5 Child Behavior Checklist. Internalizing behavior raw score at grade 5.

cbcl_ext_verh_rw_t5 Child Behavior Checklist. Externalizing behavior raw score at grade 5.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 4806 rows and 21 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1602 rows and 22 columns.

Details

`reading_raw` is the raw data in long data format.

`reading_latent` is based on `reading_raw` but the variable `dv` was log-transformed and standardized. The data set only includes measurements at grade 1, 2 and 4.

`reading_manifest` is based on `reading_latent` and was transposed to wide format introducing the three additional variables: `gaze_dur`, `tv_dur` and `ini_fix_dur`. The names for the variables were taken from `indicator` and the values were taken from `dv`.

`regression_coefficients` *Reshape a fit's pi-layer coefficients into contrast-ready arrays*

Description

Pipeline stage 2: a pure reshape of [model_parameters()]'s output (it never re-reads the 'lavaan' object itself) into the arrays the contrast math needs. Which parameter belongs in which cell is resolved through [regression_labels()]. That is the same function that generated the labels when the syntax was written, so construction and read-back cannot drift apart.

Usage

```
regression_coefficients(fit)
```

Arguments

fit	An [Lgc] object.

Value

A list with:

'B_array' '(covariate + 1) x pi x group' array of point estimates; the first covariate slice is the pi intercepts.

'B_vec' 'as.vector(B_array)', named by parameter label.

'B_vec_vcov' Sampling covariance matrix of 'B_vec'.

'B_var_vec' Between-person (co)variance of the pi's, laid out over the same vec positions, the interindividual-differences matrix used for SD bands. Zero wherever a position is not a pi intercept, or two positions are in different groups.

'labels' The label array 'B_array' was filled from.

---

resid_set *Select residual terms to constrain*

Description

Names a set of residual variances or covariances for [semnova()] to constrain, by indicator and by condition. Both are optional: omit 'indicator' and the set covers every indicator, omit the conditions and it covers every occasion.

The set is the unit of constraint. For 'resid_var' its members' residual variances are equated; for 'resid_cov' residual covariances are added among all of its members, equal to each other or freely estimated according to 'equal'.

Print a [ResidSet].

Usage

```
resid_set(indicator = NULL, ..., equal = NULL)

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

Arguments

indicator	Character vector of indicator names, or 'NULL' (the default) for every indicator. Indicators are the levels of the 'indicator' column given to [semnova()]; for a manifest dependent variable there is only one, the dv itself.
...	Ignored.
equal	Logical, or 'NULL' (the default) to let the argument the set is passed to decide. 'resid_var' sets are equated, which is the only reason to name one. 'resid_cov' sets are freely estimated. Set it explicitly to override, most usefully 'equal = TRUE' on a covariance set.
x	A [ResidSet].

Value

A [ResidSet] object.

Examples

```
# one indicator's residual variances, equal across every occasion
resid_set("fix_count")

# ... restricted to landolt sentences
resid_set("fix_count", sentence = "landolt")

# every indicator, at one occasion
resid_set(grade = "grade1")
```

`ResidSet` *ResidSet: a set of residual terms to constrain*

Description

Pure data container: fields only, no behaviour. Describes *which* residual terms a constraint applies to, in the design's own vocabulary: indicators, and the conditions they were measured in. Construct with [resid_set()], which validates; resolving the set to actual model variables happens in [semnova()], the only place that knows the design.

Public fields

`indicator` Character vector of indicator names, or NULL for every indicator.

`conditions` Named list, within-factor -> the levels the set is restricted to. Empty for every condition.

`equal` Logical. Constrain the set's terms to be equal.

Methods

Public methods:

- `ResidSet$new()`
- `ResidSet$clone()`

Method `new()`**:** Construct a ResidSet. Prefer [resid_set()].

Usage:

```
ResidSet$new(...)
```

Arguments:

`...` Fields to set, by name.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ResidSet$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`semnova` *Fit a latent repeated-measures ANOVA*

Description

The ANOVA-style entry point: describe the design in the vocabulary of the experiment (a dependent variable, within- and between-subject factors, optional indicators for a latent DV and covariates) and the model is derived from it, contrast matrices from R's own contrast machinery, and one hypothesis per model term (main effects and interactions), each testable through [test_hypothesis()].

Usage

```
semnova(
  data,
  id,
  dv,
  indicator = NULL,
  withins = NULL,
  betweens = NULL,
  covariates = NULL,
  contrasts_arg = list(),
  normalize_contrasts = FALSE,
  resid_cov = FALSE,
  resid_var = FALSE,
  invariance_within = "strong",
  invariance_between = "strong",
  sphericity = FALSE,
  compound_symmetry = FALSE,
  variance_homogeneity = FALSE,
  group_weights = c("fixed", "stochastic"),
  lv_scaling = c("effect", "referent"),
  ...
)
```

Arguments

`data`	A data.frame in **long** format: one row per case x occasion x indicator.
`id`	Character(1). Column identifying cases.
`dv`	Character(1). Column holding the dependent variable.
`indicator`	Character(1) or NULL. Column naming which indicator each row holds. 'NULL' means a manifest (single-indicator) DV.

`withins` Character vector of within-subject factor columns.

`betweens` Character vector of between-subject factor columns.

`covariates` Character vector of covariate columns, numeric, or categorical and coded into contrast columns for you (a ‘k‘-level factor becomes one ‘k - 1‘ degree-of-freedom term, coded with ‘contrasts_arg[[name]]‘ or ‘contr.treatment‘), or a named list mapping a latent covariate to its indicator columns.

`contrasts_arg` Named list, factor name -> contrast function name (e.g. ‘list(grade = "contr.sum")‘). Defaults: ‘contr.poly‘ for within factors, ‘contr.treatment‘ for between factors.

`normalize_contrasts`
Logical; scale contrast columns to unit length.

`resid_cov` Residual covariances to add, described by [resid_set()]: which indicators, in which conditions, and whether the covariances in the set are equal to each other (‘equal = TRUE‘) or freely estimated (the default). Give one set, a list of them, or ‘TRUE‘ for one set covering everything. A bare character vector of indicator names is shorthand for ‘resid_set()‘ over all conditions. ‘FALSE‘ (the default) adds none.

`resid_var` Residual variances to constrain equal, described the same way, ‘resid_set("fix_count")‘ equates that indicator’s residual variances across occasions, ‘resid_set("fix_count", sentence = "landolt")‘ only within landolt sentences, ‘TRUE‘ equates every residual variance in the model. ‘FALSE‘ (the default) leaves them all free. Ignored, with a warning, when ‘invariance_within‘ or ‘invariance_between‘ is ‘"strict"‘: that already equates residual variances, and a set would override the invariance constraint rather than add to it. A set covering fewer than two residual terms (a single occasion, say) has nothing to constrain and is dropped with a warning.

`invariance_within`, `invariance_between`
One of ‘"configural"‘, ‘"weak"‘, ‘"strong"‘, ‘"strict"‘.

`sphericity` Which model terms to constrain to sphericity, equal variances and zero covariances among the contrast variables that term spans, the structure classical repeated-measures ANOVA assumes. ‘TRUE‘ constrains every term that spans more than one contrast variable; otherwise give term names as they appear in the ANOVA table (‘"grade"‘, ‘"grade:sentence"‘, or a list of them), or an interaction as its factors (‘c("grade", "sentence")‘). ‘FALSE‘, the default, constrains nothing. Raw contrast-variable index sets, as [lgc()] takes them, are also accepted.

`compound_symmetry`
Logical. Structure the eta layer instead: one variance for every occasion and one covariance for every pair, with the contrast variables’ own (co)variances fixed to zero. Takes precedence over ‘sphericity‘, which is then ignored with a warning.

`variance_homogeneity`
Logical. Constrain those (co)variances to be equal across groups.

`group_weights` One of ‘"fixed"‘ (default, group sizes are what they are) or ‘"stochastic"‘ (group membership is random, so each group’s frequency is estimated and reported as a ‘relfreq*‘ definition).

`lv_scaling` One of ‘"effect"‘ or ‘"referent"‘.

`...` Passed to [lavaan::sem()].

Value

A [Semnova] object.

`Semnova` *Semnova: a fitted latent repeated-measures ANOVA*

Description

Pure data container, extending [Lgc] with the parsed factorial design. Fields only, no behaviour, every operation on a fit is a plain function taking it as an argument ([model_parameters()], [contrast_estimates()], [test_hypothesis()], ...). Construct with [semnova()].

Super class

`semnova::Lgc` -> `Semnova`

Public fields

`within_design` Design list from [parse_design()].

`between_design` Design list from [parse_design()].

`id, dv, indicator` Column names from the original long data.

`dv` Dependent-variable column.

`indicator` Indicator column, or NULL for a manifest DV.

`withins, betweens` The factor columns of each design.

`betweens` Between-subject factor columns.

`data_long` The original long-format data.

Active bindings

`id, dv, indicator` Column names from the original long data.

`withins, betweens` The factor columns of each design.

Methods

Public methods:

- `Semnova$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Semnova$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

strip_construct_structure *Strip a construct's own mean and (co)variance lines from measurement syntax*

Description

[build_measurement_syntax()] (via ‘semTools::measEq.syntax()‘, or its own hand-rolled branch) always leaves each construct's own latent mean and (co)variance free, correct when the measurement syntax is used standalone, but redundant once a pi layer is added: ‘pi =~ eta‘ with fixed loadings is an invertible transform, so the pi intercepts and pi (co)variances already carry the etas' entire mean and covariance structure. Leaving both free produces a singular (non-positive-definite) parameter covariance matrix.

Removes lines where a construct appears on the left of ‘~‘ (its latent mean) or ‘~~‘ (its (co)variances); measurement lines (‘=~‘) are kept. [build_pi_layer_syntax()] emits the corresponding ‘eta ~ 0*1‘ and ‘eta ~~ 0*eta‘ fixings, so the combined syntax has exactly one specification per parameter and no "modifier ... overwritten" warnings.

Usage

```
strip_construct_structure(syntax, construct_names)
```

Arguments

`syntax`	Character(1). Lavaan syntax, e.g. from [build_measurement_syntax()].
`construct_names`	Character vector of construct (eta) names whose own mean/(co)variance lines should be removed.

Value

Character(1), ‘syntax‘ with matching lines removed.

summary.Lgc *Summarize a fitted model*

Description

Summarize a fitted model

Usage

```
## S3 method for class 'Lgc'
summary(object, level = 0.95, ...)
```

Arguments

`object`	An [Lgc] object.
`level`	Confidence level for the interval columns.
`...`	Unused.

Value

An object of class ‘semnova_summary‘.

`test_hypothesis` *Test a hypothesis on a fitted model*

Description

Alias for [contrast_estimates()]. The same object is both "the estimates for this contrast" and "the test of this hypothesis", so both names are accurate and neither wraps extra logic.

Usage

```
test_hypothesis(fit, hypothesis, level = 0.95)
```

Arguments

`fit`	An [Lgc] object.
`hypothesis`	A [Hypothesis] object (see [hypothesis()]).
`level`	Confidence level for the interval columns.

Value

An object of class ‘semnova_contrast‘.

`validate_hypothesis` *Validate and default-fill a Hypothesis against a fit*

Description

Fills in default ‘M‘/‘G‘/‘L‘/‘K‘ when they’re omittable (a single eta, a single group, or no covariates respectively) and checks every supplied matrix’s dimensions against ‘fit‘’s public fields (‘fit$etas_mmodel‘/‘fit$group_labels‘/‘fit$covariates_mmodel‘) – errors with an informative message otherwise. Mutates ‘hypothesis‘ in place (R6 objects are reference-semantics) and returns it, so repeated validation of an already-filled hypothesis is a cheap no-op dimension check.

Usage

```
validate_hypothesis(hypothesis, fit)
```

Arguments

`hypothesis`	A [Hypothesis] object.
`fit`	An object exposing ‘etas_mmodel‘, ‘group_labels‘, ‘covariates_mmodel‘ as public fields (an ‘Lgc‘/‘Semnova‘ fit, or anything with the same shape, e.g. a plain list, for testing).

Value

‘hypothesis‘, invisibly mutated.

Index

∗ **datasets**
    reading_raw, 26

anova.Lgc *(*anova_table*)*, 2
anova_table, 2

build_covariate_syntax, 3
build_group_weight_syntax, 4
build_measurement_syntax, 5
build_pi_layer_syntax, 6
build_resid_cov_syntax, 7
build_resid_var_syntax, 8

contrast_estimates, 8
contrast_matrix, 9
contrast_table, 10

dag, 10
dag_save, 12

expand_categorical_covariates, 13

fit_measures, 14

generate_anova_hypotheses, 15

Hypothesis, 16
hypothesis, 16

identification_scheme, 18

Lgc, 20
lgc, 18

model_parameters, 21

n_occasions *(*param-names*)*, 22

param-names, 22
param_label *(*param-names*)*, 22
parse_design, 24
parse_param_label *(*param-names*)*, 22
plot.Lgc, 25
pretty_label *(*param-names*)*, 22
print.ResidSet *(*resid_set*)*, 28

reading_latent *(*reading_raw*)*, 26
reading_manifest *(*reading_raw*)*, 26
reading_raw, 26
regression_coefficients, 27
regression_labels *(*param-names*)*, 22
resid_set, 28
ResidSet, 29

Semnova, 32
semnova, 30
semnova::Lgc, *32*
strip_construct_structure, 33
summary.Lgc, 33

test_hypothesis, 34

validate_hypothesis, 34