Package 'Gtheory4LLM'

November 2, 2026

Type Package

Title Generalizability Theory for LLM Subjective Tasks

Version 1.0.0

Author Jin Liu [aut, cre, cph]

Maintainer Jin Liu <Veronica.Liu0206@gmail.com>

Description Studies the reliability and generalizability of subjective judgments produced by large language models (LLMs), including annotation, rating, and structured LLM-as-a-judge tasks. Specifies evaluator, prompt, generation, and repeated-run facets through crossed or explicitly nested random sources with configurable item interactions. Fits univariate models, joint Gaussian models, and joint discrete models for binary, ordinal, and unordered categorical outcomes, with source-specific covariance. Gaussian models use exact balanced likelihood, with a bounded observed-row route for incomplete univariate panels. Discrete models use dense or sparse first-order Laplace approximation with Gaussian latent random effects. Supported balanced decision studies compare evaluator, prompt, and replication allocations using observed Gaussian or explicitly requested latent binary and ordinal reliability, with random or fixed facets after Brennan (2001). 'OpenMx' Gaussian fits report asymptotic Wald standard errors for variance components and delta-method coefficient intervals when curvature checks permit them; the 'lme4' route returns point estimates only. Discrete fits offer optional full-information latent coefficient intervals, subject to numerical and boundary checks. An object bootstrap refits resampled objects under each fit's acceptance rule and gives percentile and bias-corrected and accelerated intervals for components, coefficients and decision studies. Provides fixed-layout call
effects, categorical score and category-vector reliability projections, repeated-panel decision agreement, bounded cost-aware comparisons, and Gaussian or latent binary/ordinal pilot precision planning, and categorical point-estimator pilot operating characteristics. Joint
Gaussian-discrete fitting is not implemented. Includes three publicly archived LLM annotation datasets covering hate-speech, mental-health, and drug-review tasks. Discrete fitting retains conservative size limits; the preflight report describes supported designs and computational limits. Generalizability coefficients follow the variance-decomposition framework

of Brennan (2001) <doi:10.1007/978-1-4757-3456-0>.

License GPL-3

Encoding UTF-8

Depends R (>= 4.5.0)

Imports Matrix (>= 1.6.0), OpenMx (>= 2.22.11), graphics, methods, parallel, stats, tools, utils

Suggests lme4, ordinal, knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/Veronica0206/Gtheory4LLM

BugReports https://github.com/Veronica0206/Gtheory4LLM/issues

Collate 'design.R' 'batch.R' 'family.R' 'gaussian_retry.R' 'gaussian_initializer.R' 'gaussian_engine.R' 'gaussian_observed.R' 'gaussian.R' 'gaussian_lme4.R' 'discrete_response.R' 'discrete_dense.R' 'discrete_sparse.R' 'selected_inverse.R' 'discrete_sparse_envelope.R' 'discrete_gradient.R' 'discrete_boundary_escape.R' 'discrete_mode.R' 'discrete_sparse_mode.R' 'discrete.R' 'discrete_check.R' 'discrete_information.R' 'preflight.R' 'diagnostics_stages.R' 'fit.R' 'methods.R' 'reliability.R' 'batch_reliability.R' 'planning.R' 'simulation.R' 'bootstrap.R' 'discrete_simulation.R' 'categorical_targets.R' 'visualization.R' 'report.R' 'examples.R'

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-11-01 23:20:21 UTC

Contents

Gtheory4LLM-package . . . 3
gaussian_observed . . . 5
gtheory_datasets . . . 7
gtheory_drug_review . . . 10
gtheory_hate_speech . . . 11
gtheory_mental_health . . . 13
gt_batch . . . 15
gt_batch_dstudy . . . 18
gt_batch_reliability . . . 19
gt_bootstrap . . . 22
gt_categorical_target . . . 25
gt_components . . . 29
gt_control . . . 30
gt_design . . . 38
gt_diagnostics . . . 41

gt_dstudy 45
gt_example 48
gt_family 50
gt_fit 51
gt_fit_methods 56
gt_laplace_check 60
gt_pilot_plan 63
gt_plan 68
gt_preflight 73
gt_reliability 80
gt_report 85
gt_score 87
gt_simulate 88
plot.gt_reliability 91
print.summary.gt_fit 92

Index **94**

`Gtheory4LLM-package` *Generalizability Theory for LLM Subjective Tasks*

Description

Studies the reliability and generalizability of subjective judgments produced by large language models (LLMs), including annotation, rating, and structured LLM-as-a-judge tasks. Supported balanced decision studies compare projected reliability across evaluator, prompt, and repeated-run allocations.

Details

Use `gt_design()` to declare random sources, `gt_preflight()` to inspect design and resource limits, `gt_family()` to distinguish the observation family, and `gt_fit()` to estimate a model. Inspect `gt_diagnostics()` before interpreting coefficients, then use `gt_dstudy()` to compare an allocation grid with a substantively chosen reliability target.

The modeling interface supports selected item interactions, crossed and explicitly nested sources, and joint multivariate source covariances. Measurement roles are specified by the research design; facet names do not automatically impose nesting or remove item interactions. Traditional method-of-moments G theory already includes random effects and crossed/nested designs. This package integrates design-specific covariance models, outcome-appropriate likelihoods, and supported decision studies; superiority in estimation accuracy requires comparative validation.

Gaussian outcomes use exact balanced likelihood by default. A bounded univariate lme4 option fits missing whole cells and checks an independent observed-row Gaussian likelihood. Binary, ordinal, and unordered categorical outcomes use their specified observation distributions with Gaussian latent random effects, fitted by dense or sparse calculations of the same first-order Laplace approximation. Joint Gaussian models and joint discrete models are supported; a joint Gaussian-discrete likelihood is not implemented. Selecting sparse algebra does not establish full-panel performance or approximation accuracy.

Observed Gaussian reliability and latent binary, ordinal or nominal reliability are distinct estimands. A nominal outcome's latent coefficients are the mean and range of the generalized eigenvalues over its category contrasts, invariant to the reference category under an exchangeable or unstructured covariance. Ordinary analytic coefficients require a complete balanced coded panel; observed-row Gaussian fits instead require an explicit projection to an intended complete balanced allocation. Projections depend on fitted source covariances and the specified universes of evaluators and prompts. Adding arbitrarily chosen LLMs does not ensure those assumptions hold. Available delta-method intervals propagate component-estimation uncertainty under the declared model, not violations of facet exchangeability or annotation accuracy against human judgments. Coefficients alone do not select a minimum-cost allocation. Default discrete delta-method intervals require optional full observed information that passes numerical acceptance, boundary, step-stability and curvature checks; their asymptotic interpretation is conditional on first-order Laplace, with finite-sample coverage unestablished. An explicit `gt_bootstrap` instead supplies object-bootstrap intervals without observed information, conditional on the observed instrumentation levels and accepted refits.

The study-planning functions distinguish pilot item count, annotations per item and items per request. Use `gt_plan` for bounded cost-aware allocation search with explicit user-supplied prices, and `gt_batch_reliability` for fixed-layout Gaussian batch targets. `gt_categorical_target` separately defines observed-score, relabeling-invariant category-vector and plurality-decision targets by repeated panel simulation. Decision agreement is accompanied by decision diversity; perfect agreement alone does not establish discrimination. Categorical target planning holds parameters and any fitted call grouping fixed, and reports Monte Carlo integration errors rather than parameter confidence intervals. These do not infer annotation accuracy, optimal provider identities or a new batch-size effect from an existing fit.

Three publicly archived LLM annotation panels cover hate-speech, mental-health, and drug-review tasks, with eight outcome sets and fifteen response codings. The dataset overview and dedicated Hate-Speech, Mental-Health, and Drug-Review topics describe variables, category mappings, preprocessing, and source and annotation-study references. The package contains design identifiers and modeled annotations from the public OSF deposit, doi:10.17605/OSF.IO/K9CAJ.

Package code is distributed under GPL-3; real annotation data retain the CC BY 4.0 license in the public deposit's codebook. See '`DATA_LICENSE.md`'. Numerical acceptance does not establish Laplace approximation adequacy, a global optimum, or empirical validity.

The OpenMx Gaussian path can report asymptotic Wald standard errors for source variance components and delta-method intervals for G and Phi; the lme4 path does not. See `gt_reliability`. Model-based Gaussian and categorical simulation are available through `gt_simulate`.

Use `gt_pilot_plan` to study interval precision for Gaussian, binary or ordinal outcomes. Its route for categorical point estimates also supports nominal outcomes and fixed calls. It reports acceptance, availability and conditional error summaries. This route supplies no parameter intervals or interval-width criterion. These functions do not provide calibrated bootstrap intervals. `gt_bootstrap` resamples the objects of an accepted fit and gives percentile or BCa intervals for components, coefficients and D studies, conditional on the observed instrumentation levels; their coverage has been assessed only in a Gaussian pilot, where it fell below the nominal level. Discrete observed information is opt-in, and unsupported or unusable uncertainty remains explicitly unavailable rather than imputed.

Why this package declares R (>= 4.5.0): R 4.5.0 supplies the SHA-256 function used to identify modeled observations for bootstrap verification. The implementation calls `tools::sha256sum()`. Its OpenMx dependency also uses a C interface introduced in that R version. Loading the stan-

dalone functions through 'load_functions.R' has the same runtime requirement. Earlier reported checks with older R/OpenMx pairs belong to earlier source versions and do not qualify the current implementation.

The vignette "How many evaluators, prompts, and runs?" demonstrates fitting through reliability, intervals, fixed facets, and a decision study using synthetic continuous judgments. Open it with `browseVignettes()`. The installed 'doc/LLM-workflow.R' is the same workflow as a script.

Author(s)

Jin Liu (author and maintainer), <Veronica.Liu0206@gmail.com>

References

Brennan, R. L. (2001). *Generalizability Theory*. Springer. doi:10.1007/9781475734560.

Jiang, Z., Raymond, M., Shi, D., and DiStefano, C. (2020). Using a linear mixed-effect model framework to estimate multivariate generalizability theory parameters in R. *Behavior Research Methods*, 52, 2383–2393. doi:10.3758/s1342802001399z.

See Also

gt_design, gt_preflight, gt_fit, gt_dstudy, gt_example, glm

`gaussian_observed` *Bounded Gaussian Fitting with Missing Whole Cells*

Description

An explicit observed-row option fits one Gaussian outcome through lme4 and checks its ML or REML criterion against an independent dense covariance likelihood.

Details

Request `gt_control(gaussian = list(engine = "lme4", panel = "observed"))`. The default `panel = "complete"` retains the balanced factorial-contrast workflow. The observed-row option does not impute omitted cells, replace missing outcomes, aggregate repeated observations, or alter the declared random sources. Each retained full cell must contain exactly one finite numeric outcome. The object and every instrumentation facet must have at least two observed levels. Joint outcomes, unequal replication, within-cell replication and declared batch effects are not supported by this option.

For observed rows, the model is

$$y = X\beta + \sum_s Z_s u_s + \epsilon, \qquad V = \sum_s \sigma_s^2 Z_s Z_s' + \sigma_e^2 I.$$

There is one fixed intercept. Random-source grouping matrices follow the declared source terms. ML profiles the intercept; REML additionally includes $\log|X'V^{-1}X|$ using the unscaled intercept

column, matching lme4's criterion. The check uses only retained rows and independent dense Cholesky calculations, while lme4 uses its mixed-model representation.

The source and residual covariance kernels must be linearly independent in the observation space after removing the intercept. A failed or numerically near-dependent kernel check is a refusal before fitting. This algebraic check does not establish adequate sample size, precision, recovery or interval coverage. Numerical acceptance still requires optimizer completion without convergence warnings, constrained stationarity, a reproducing restart, and agreement with the independent likelihood at the fitted parameters. The lme4 path does not compute source-variance standard errors or coefficient intervals.

Resources. `max_observed_rows = 1000` and `max_preparation_bytes = 512 * 1024^2` bound the dense reference. The allocation estimate includes covariance kernels and identification working space, but excludes caller data and lme4 objects. Changing these limits does not establish validated behavior at the larger size.

Interpretation. Likelihood inference assumes the omitted whole cells are ignorable for the declared Gaussian model. The software cannot establish this assumption from the retained rows. In particular, omission depending on unobserved outcomes can bias component estimates. Levels never observed cannot be recovered by declaring them in a factor.

Fitting an incomplete panel does not supply analytic reliability for its irregular observed allocation. Ordinary `gt_reliability()` remains unavailable. A separately requested balanced D-study projection describes an intended complete allocation in the same facet universes, assuming the same source variances. It is not a coefficient for the incomplete observations. The fit records this distinction and its assumptions in `projection`, and records `design$panel_type = "observed"`. Its prepared metadata retain resource and kernel-identification diagnostics, without another copy of the raw outcomes or dense kernels.

See Also

`gt_fit`, `gt_control`, `gt_dstudy`

Examples

```
if (requireNamespace("lme4", quietly = TRUE)) {
  set.seed(1317)
  d <- expand.grid(item = 1:24, rater = 1:5)
  d$y <- 8 + rnorm(24, sd = 2)[d$item] +
    c(-2, -.8, .1, 1, 2.1)[d$rater] + rnorm(nrow(d), sd = .7)
  observed <- d[-seq(4L, nrow(d), by = 7L), ]
  fit <- gt_fit(observed, y ~ item + rater, measured = "item",
    control = gt_control(gaussian = list(engine = "lme4", panel = "observed")))
  gt_components(fit)
  gt_diagnostics(fit)
}
```

gtheory_datasets *Three LLM Annotation Datasets: Design, Coding, and Sources*

Description

Documents the Hate-Speech, Mental-Health, and Drug-Review modeling panels distributed with Gtheory4LLM. The three underlying datasets provide eight outcome sets and fifteen combinations of outcome set and coding through `gt_example`.

Format

Calling `gt_example(name)` returns a list. Its `data` element is a data frame containing the five shared factor columns below and the selected outcome column(s). The other elements identify the outcomes, observation families, object and facets, coding, source, and interpretation notes. Installed resources also include `source_provenance`.

item 100 text-object identifiers retained from the corresponding public annotation table.

evaluator Four stored run identifiers:

- `gemini-2.5-flash`
- `gpt-4o-mini`
- `llama-70b` (Llama-3.3-70B)
- `mistral-small` (Mistral-Small-24B)

prompt Three prompt identifiers: `cot`, `minimal`, and `rubric`. The `cot` condition adds a structured reasoning checklist.

temp Six temperature levels: `0`, `0.2`, `0.4`, `0.6`, `0.8`, and `1`. They are stored as factor levels.

seed Three replicate-run identifiers: `1001`, `2001`, and `3001`. A repeated seed does not guarantee identical provider output.

Details

Each dataset contains a 100-item crossed LLM annotation panel from the public OSF annotation deposit (Liu, 2026). Task definitions and annotation protocols are described in the three task-specific studies cited in the dedicated dataset topics. HateXplain, Sentiment Analysis for Mental Health, and Drugs.com reviews supplied the original text material.

Items were selected using screening-pass entropy stratification. Each of the 100 items was evaluated under all combinations of four evaluators, three prompt types, six temperatures, and three seeds, yielding 216 rows per item and 21,600 rows per outcome set. These are repeated measurements on 100 text objects; the row count is not the number of independent texts. The selected items and evaluator panel define the scope of these examples.

The installed resources contain design identifiers and modeled LLM outcomes. The source study's parsing and post-processing are retained, including default labels for omitted items within successfully parsed batches. There are no missing modeling values in the stored panels; this does not establish that every original API response supplied a usable item annotation. Raw texts, prompt text, original corpus reference labels, and API metadata are outside these minimal modeling tables.

Temperature, seed, and the intended universe

Temperature is a controlled setting. Whether it is treated as fixed or random in a reliability analysis depends on the intended universe of generalization. If inference concerns exactly the observed temperature settings, an analyst can declare `gt_reliability(fit, fixed = "temp")`. Generalizing beyond those settings instead requires an explicit population and exchangeability assumptions; selection by the researcher alone does not determine the choice.

The observed agreement among seed replicates changes across temperature. The table gives the proportion of item-by-evaluator-by-prompt cells in which all three seeds return the same label:

temperature	0	0.2	0.4	0.6	0.8	1
Hate-Speech	0.915	0.862	0.837	0.817	0.780	0.733
Mental-Health (nominal)	0.914	0.842	0.835	0.789	0.749	0.678
Drug-Review	0.840	0.767	0.711	0.686	0.631	0.581

These are descriptive observed-label agreement rates. Their differences motivate checking the response and covariance model, but do not by themselves establish that a latent seed variance is heterogeneous. For discrete outcomes, changes in category probabilities can change observed agreement even with constant latent random-effect variance. Agreement is below one at temperature 0, so that setting does not guarantee identical recorded labels across runs.

Declaring `fixed = "temp"` changes the reliability estimand after fitting; it does not change the G-study likelihood or correct covariance misspecification. A scientifically defined analysis within one temperature is one possible sensitivity analysis. In that case, omit the now-constant temperature column from the fitted facet declaration while recording the conditioned setting in the analysis. A model explicitly allowing variation by temperature is another possible approach, provided its implementation and assumptions are justified. Neither treatment is selected automatically from this descriptive table.

Dataset topics and outcome sets

Hate-Speech One ordinal outcome set. See the Hate-Speech data topic for category definitions and the original corpus and annotation-study references.

Mental-Health Five outcome sets. See the Mental-Health data topic for working-score mappings, binary flags, unordered labels, and source references.

Drug-Review Two ordinal outcome sets. See the Drug-Review data topic for holistic sentiment, four aspect definitions, and source references.

Coding and use

The default `coding = "native"` preserves the binary, ordinal, or unordered categorical endpoint where defined. The Mental-Health 7L and 3L variants remain explicitly Gaussian working scores under both coding options; they are not established clinical severity scales. Seven outcome sets also support `coding = "numeric"`, which reproduces the original Gaussian working-score analysis. The nominal Mental-Health set has native coding only. The eight sets are alternative views of three panels, not eight independent datasets.

Use `gt_example()` to list the catalog and `gt_example(name)$data` to access a table. Each panel has 21,600 rows, so every native discrete coding exceeds the dense discrete engine's default limits

by more than an order of magnitude, and raising the limits does not make the dense engine practical at that size. These are data resources, not full-panel discrete fitting examples: use `gt_preflight()` before requesting a discrete fit, and do not remove item interactions merely to obtain an accepted fit. Discrete demonstrations require a scientifically specified small panel and supported source structure; successful loading alone does not establish fitting feasibility. Numerical model assumptions and coefficient scales are described in `gt_fit` and `gt_reliability`.

Source

Liu's public OSF annotation deposit:

doi:10.17605/OSF.IO/K9CAJ.

The source files are:

- 'hate_labeling_final.csv'
- 'mh_labeling_final.csv'
- 'drug_labeling_final.csv'

Their checksums match the public deposit. Package resources retain only design identifiers and modeled annotations; category types, working-score mappings, and grouped flags are defined by `gt_example()`.

The deposit's data codebook licenses derived artifacts under CC BY 4.0. The installed supporting files are:

- 'DATA_LICENSE.md': attribution and modification details.
- 'extdata/manifest.csv': source and resource checksums.
- 'DATASET_REFERENCES.bib': the complete task bibliography.

References

Liu, J. (2026). *LLM Annotation Reliability: A Generalizability Theory Analysis of Hate Speech, Mental Health, and Drug Review Tasks*. Open Science Framework, public annotation dataset.

doi:10.17605/OSF.IO/K9CAJ.

The three dedicated dataset topics provide the original corpus and related annotation-study references.

See Also

`gt_example`, Hate-Speech data, Mental-Health data, Drug-Review data

Examples

```
gt_example()
x <- gt_example("hate_speech")
dim(x$data)
vapply(x$data[x$facets], nlevels, integer(1))
x$outcomes
system.file("DATASET_REFERENCES.bib", package = "Gtheory4LLM")
```

gtheory_drug_review *LLM Drug-Review Sentiment and Four Aspect Annotations*

Description

Two outcome sets from repeated LLM judgments of 100 medication reviews: five-category holistic sentiment and four three-category aspects. The modeling tables are accessed with `gt_example`.

Format

Each call returns a list whose `data` element has 21,600 rows. The five shared factor columns are `item`, `evaluator`, `prompt`, `temp`, and `seed`: 100 reviews crossed with four evaluators, three prompts, six temperatures, and three replicate seeds. Their stored levels are documented in the dataset overview.

drug_review One `score` column. Native coding is an ordered factor: 1 = VERY_NEGATIVE, 2 = NEGATIVE, 3 = NEUTRAL, 4 = POSITIVE, 5 = VERY_POSITIVE. These are holistic sentiment categories, even though the source CSV calls its score column `severity`.

drug_review_4aspect Four outcome columns: `efficacy`, `safety`, `burden`, and `cost`. Each is an ordered factor with levels NEGATIVE, NEUTRAL, POSITIVE under native coding.

The native families are ordinal with probit links. For either set, `coding = "numeric"` returns numerical Gaussian working scores: 1–5 for holistic sentiment and 1–3 for each aspect. The loader also returns outcome names, families, design roles, coding, source, and notes; installed resources include `source_provenance`.

Details

The aspect annotations summarize the review's sentiment about:

efficacy Whether the treatment helped or worked (positive) or did not work or worsened the problem (negative).

safety Tolerability or absence of side effects (positive) versus side effects or adverse events (negative).

burden Convenience or ease of use (positive) versus inconvenience, adherence difficulty, or regimen complexity (negative).

cost Affordability or coverage (positive) versus expense, coverage denial, or high out-of-pocket costs (negative).

The study's rubric assigned NEUTRAL when an aspect was unclear, unmentioned, or mixed. This category should not uniformly be read as an explicitly neutral patient opinion. Aspects can disagree with the holistic sentiment response.

These are LLM-generated annotations of reviews drawn from the UCI Drug Review corpus, collected from Drugs.com. They are not the original patient-provided 1–10 ratings. The installed tables omit review text, medication and condition metadata, source ratings, probability vectors, and API metadata.

Every modeling row is complete after the source study's preprocessing, which allowed NEUTRAL defaults for items omitted from an otherwise parsed response batch. The package preserves these stored annotations and does not identify defaults separately. The two sets share the same sampled reviews and design; they are not independent datasets. The complete discrete tables exceed current dense backend limits; the examples below only load and summarize them.

Source

Derived from 'drug_labeling_final.csv' in the public OSF annotation deposit (Liu, 2026), doi:10.17605/OSF.IO/K9CAJ. Gräßer et al. (2018) describe the source drug-review data; Liu (2026) describes the related aspect-level LLM annotation study. The common G-theory source is cited in the dataset overview.

References

Gräßer, F., Kallumadi, S., Malberg, H., and Zaunseder, S. (2018). Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning. *Proceedings of the 2018 International Conference on Digital Health*, 121–125. doi:10.1145/3194658.3194677.

Liu, J. (2026). Learning to Fuse Aspect-Level LLM Annotations for Low-Quality Ordinal Sentiment Supervision. *SSRN working paper*, 6244519; revised June 17, 2026.

doi:10.2139/ssrn.6244519.

See Also

Dataset overview, `gt_example`, `gt_family`

Examples

```
holistic <- gt_example("drug_review")
str(holistic$data$score)
table(holistic$data$score)

aspects <- gt_example("drug_review_4aspect")
str(aspects$data[aspects$outcomes])
table(aspects$data$efficacy, aspects$data$safety)
```

`gtheory_hate_speech` *LLM Hate-Speech Annotations of HateXplain Texts*

Description

An ordinal outcome set containing repeated LLM judgments of 100 HateXplain text items. The installed package includes the modeling table and source provenance, accessed through `gt_example`.

Format

The loader returns a list; its `data` element is a data frame with 21,600 rows and six columns. Each item has 216 measurements in the complete item-by-evaluator-by-prompt-by-temperature-by-seed panel.

item Factor identifying the 100 text items.

evaluator, prompt, temp, seed Factors identifying the four evaluators, three prompt conditions, six temperatures, and three replicate seeds. Their stored levels and interpretation are documented in the dataset overview.

score With `coding = "native"`, an ordered factor with levels `1`, `2`, `3`: NORMAL, OFFENSIVE, and HATE, respectively. With `coding = "numeric"`, the same values are numeric Gaussian working scores.

The remaining list elements identify the outcome, family, object, facets, coding, source, and interpretation notes; installed resources also provide `source_provenance`.

Details

The native family is ordinal with a probit link. The category order expresses the study's normal/offensive/hate distinction; using the numeric scores additionally treats the category steps as equally spaced.

These outcomes are the repeated LLM annotations used in the G-theory study, not the original HateXplain crowdworker labels. The data table contains design identifiers and the modeled outcome only; original posts, crowdworker labels, probability vectors, prompts, and API metadata are not included.

Every modeling row has a recorded value after the original study's preprocessing. That pipeline could assign NORMAL to items omitted from an otherwise parsed response batch. The package preserves the stored outcomes and does not separately identify such defaults. Completeness of the modeling table therefore does not imply that every original response was complete.

The full table exceeds the current dense discrete fitting limits. The examples below describe the data without fitting the full panel; see the dataset overview for the shared design and fitting scope.

Source

Derived from '`hate_labeling_final.csv`' in the public OSF annotation deposit (Liu, 2026), doi:10.17605/OSF.IO/K9CAJ. HateXplain is the original text corpus (Mathew et al., 2021); Liu (2026) describes the related rubric-conditioned annotation study. These references concern corpus and annotation provenance, respectively. The common G-theory source is cited in the dataset overview.

References

Mathew, B., Saha, P., Yimam, S. M., Biemann, C., Goyal, P., and Mukherjee, A. (2021). HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17), 14867–14875. doi:10.1609/aaai.v35i17.17745.

Liu, J. (2026). Rubric-conditioned large language model labeling: Agreement, uncertainty, and label consistency in subjective text annotation. *Computers in Human Behavior*, 181, 108988. doi:10.1016/j.chb.2026.108988.

See Also

Dataset overview, gt_example, gt_family

Examples

```
hate <- gt_example("hate_speech")
dim(hate$data)
str(hate$data)
table(hate$data$score)

# The historical numeric coding represents the same stored judgments.
working <- gt_example("hate_speech", coding = "numeric")
str(working$data$score)
```

---

gtheory_mental_health *LLM Mental-Health Text Annotations in Five Outcome Sets*

Description

Five representations of repeated LLM annotations of the same 100 text items from the Sentiment Analysis for Mental Health corpus: unordered labels, six binary flags, three binary groups, and two historical working-score codings. Load each modeling table with gt_example.

Format

Each call returns a list whose data element has 21,600 rows. The five shared factor columns are item, evaluator, prompt, temp, and seed: 100 items crossed with four evaluators, three prompts, six temperatures, and three replicate seeds. Stored factor levels are documented in the dataset overview. Additional columns depend on the requested set:

mental_health_nominal One label column, an unordered factor with levels NORMAL, STRESS, ANXIETY, DEPRESSION, BIPOLAR, PERSONALITY_DISORDER, and SUICIDAL. The categorical family uses a softmax link and NORMAL as reference. Only coding = "native" is available.

mental_health_6flag Six integer 0/1 columns:

- depression
- anxiety
- suicidal
- stress
- bipolar
- personality_disorder

A value of one indicates the corresponding LLM-assigned evidence flag. Native families are binary with probit links.

mental_health_3group Three integer 0/1 columns:

- stress: the original stress flag.

- `anxiety_depression`: anxiety OR depression.
- `bipolar_personality_suicidal`: the logical OR of bipolar, personality-disorder, and suicidal flags.

The groups may co-occur; they are not mutually exclusive categories. Native families are binary with probit links.

mental_health_7L One integer `score` column: NORMAL = 1, STRESS = 2, ANXIETY = 3, DEPRESSION = 4, BIPOLAR = 5, PERSONALITY_DISORDER = 6, SUICIDAL = 7. This is a Gaussian working score under both coding options.

mental_health_3L One integer `score` column: NORMAL or STRESS = 1; ANXIETY or DEPRESSION = 2; BIPOLAR, PERSONALITY_DISORDER, or SUICIDAL = 3. This is a Gaussian working score under both coding options.

The loader also returns outcome names, families, design roles, coding, source, and interpretation notes; installed resources include `source_provenance`.

Details

The 7L and 3L codings reproduce study-defined numerical analyses. They do not establish a clinical severity order or equal distances between mental-health categories. The nominal set preserves the unordered label identity. The six flags describe requested textual evidence and are not clinical diagnoses or a set of mutually exclusive classifications. The three-group binary set differs from the single-outcome 3L working score.

For the six-flag and three-group sets, `coding = "numeric"` retains the same 0/1 values but declares Gaussian working-score families. The two working-score sets remain Gaussian even with `coding = "native"`.

All sets reuse LLM-generated annotations of the same sampled items, not independent samples or the original corpus's reference status labels. Original texts, corpus reference labels, probability vectors, and API metadata are not bundled. The stored tables have no missing modeling values after the original preprocessing, which allowed NORMAL defaults for items omitted from an otherwise parsed batch. Defaults are not separately identified in these modeling resources. See the dataset overview for common provenance and the full-panel discrete fitting limit.

Source

Derived from '`mh_labeling_final.csv`' in the public OSF annotation deposit (Liu, 2026).

doi:10.17605/OSF.IO/K9CAJ.

Sarkar's corpus is the original text source. Liu's public annotation study describes the associated multi-view labeling work; its current title is listed below. The common G-theory source is cited in the dataset overview.

References

Sarkar, S. (n.d.). *Sentiment Analysis for Mental Health*. Kaggle dataset; accessed September 12, 2026. The original source URL is retained in the installed task bibliography.

Liu, J. (2025). LLM-Based Annotation as Multi-View Supervision for Affective Text Classification: Soft Labels, Multi-Task Decomposition, and Entropy Diagnostics. *SSRN working paper*, 5964738; revised March 19, 2026. doi:10.2139/ssrn.5964738.

See Also

Dataset overview, gt_example, gt_family, gt_score

Examples

```
nominal <- gt_example("mental_health_nominal")
str(nominal$data$label)
table(nominal$data$label)

flags <- gt_example("mental_health_6flag")
str(flags$data[flags$outcomes])
table(flags$data$depression, flags$data$anxiety)

groups <- gt_example("mental_health_3group")
str(groups$data[groups$outcomes])
working <- gt_example("mental_health_3L")
table(working$data$score)
```

---

gt_batch *Declare That Items Were Annotated Several to a Call*

Description

Declares that the object of measurement was submitted in calls of several items, so that items sharing a call are not independent. Without a declaration every function treats items as independent given the declared sources.

Usage

```
gt_batch(size, order = NULL, by = NULL, sequential = FALSE,
  neighbor = FALSE, id = NULL)
## S3 method for class 'gt_batch'
print(x, ...)
```

Arguments

size	The number of items in a call: one integer of at least 2. With id, the most a call may hold.
order	Optional name of a numeric column giving the order in which items were submitted. Without id, each item has one distinct value, the same in every condition, and NULL uses the order in which items first appear in the data. With id, it gives the position of each row's item in its call, and NULL uses the stored order of the rows of each call.
by	Optional name of an instrumentation facet across whose levels the dependence may differ, for example the evaluator.
sequential	FALSE, or how many preceding items may affect an item: a whole number from 1 to size - 1. TRUE means every preceding item in the call.

neighbor	FALSE, or how many positions apart two items may be and still affect each other, in either direction: a whole number from 1 to size - 1. TRUE means any distance within the call.
id	Optional name of a column identifying the call that produced each row. NULL, the default, infers the batches from the item order.
x	A gt_batch object for the print method.
...	Further print arguments; currently unused.

Details

A declaration is passed to gt_design and travels with the design to every function that receives it or a fit made from it.

By default items in one call are assumed to affect each other equally, whatever their order, and items in different calls are independent given the declared sources. That the dependence varies with position is a separate, optional claim, and two patterns fit it. In a sequential pattern an item is affected by items submitted before it. In a neighbour pattern items close together affect each other in either direction. Data in which every item keeps the same position cannot always tell the two apart, so each is declared on its own, and declaring one asks for it to be examined rather than asserting it.

Inferred and recorded calls. Which items shared a call is either inferred or recorded. Without id it is inferred: the items present in the data are cut into batches of size in their order, the last holding what remains, and a batch is the same items, in the same order, in every condition. That is all that data without a call column allow, and it is a reconstruction. An item removed after collection moves every later item into another position and possibly another batch, and nothing in the data shows it; calls of other sizes and regrouping between conditions cannot be seen either. With id the calls are read from the data. They may then hold different items in different conditions and fewer items than size; a call may not span conditions, hold an item twice, or hold more than size items. Record the call when collecting data, and prefer id whenever that record exists.

Recorded calls are still read from the rows supplied, and three things follow. A call is counted by the items of it that are present, so a call that was sent size items and lost one afterwards looks the same as a call sent one item short. A call with no row left is not counted at all. And a position is a rank among the items present in a call: an order column holding 1, 3 and 4 gives positions 1, 2 and 3. Positions are the positions at which items were sent only when nothing was removed after collection, so a distance between two positions is not yet a distance in the call as it was sent. Keep the unfiltered rows, or the number of items sent in each call, if that distinction will matter.

What a fit does with a declaration. Gaussian and discrete fits model the declaration when the batches form a complete factorial with the facets: the same batches in every condition, every batch at the declared size, at least two batches, and one row in each cell. They estimate one shared call effect, a random effect for each batch under each condition that its items share equally. The effect is reported as the source Call beside the declared sources; with several outcomes it has a source covariance matrix structured through covariance. A Gaussian fit uses its exact balanced likelihood, in which an item is a batch-by-slot cell. A binary, ordinal, or unordered categorical fit uses first-order Laplace marginal ML, through the selected dense or sparse backend. Which slot an item occupies has no effect of its own. Discrete standard errors require an explicitly requested, accepted observed-information calculation; boundary or unstable curvature leaves them unavailable.

Unsupported layouts are recorded and audited but not modelled: batches of unequal size, batches that differ between conditions, more than one row in a cell, or a single batch. The fit then does not

include a shared-call source and records the reason. `sequential`, `neighbor` and `by` are recorded and not yet estimated. `gt_preflight` reports beforehand which treatment a fit will use. Each modelled call adds latent random coordinates, so the selected engine's resource guards still apply.

Coefficients. A call effect is shared by the items of one call and by no others, so its part in relative error depends on which items share calls. The ordinary `gt_reliability` and `gt_dstudy` weights do not encode that layout and refuse such a fit. Use `gt_batch_reliability` or `gt_batch_dstudy` for explicit item-score, contrast or aggregate targets conditional on the fitted equal fixed Gaussian layout. Those projections are point estimates and do not generalize to regrouped items or a new batch size. Discrete call effects can be simulated under the fitted item-to-batch membership; this does not supply a default analytic latent coefficient for that layout.

Every result says what happened to a declaration. Fits and diagnostics print the status, the data-frame exports of coefficients carry it as `batch_status`, and `gt_report` records it. It takes one of three values:

- `"not_declared"`: the design declares no batch.
- `"declared_not_modelled"`: items in one call are treated as independent, for the reason given.
- `"modelled"`: the fit estimates the shared call effect.

Value

A `gt_batch` list holding the size, the order column, the facet named in `by`, the two reaches as integers with 0 for a pattern that is off, the call column, and the assumption they state in words. `print()` shows the declaration and that assumption, and returns the object invisibly.

See Also

`gt_design`, `gt_preflight`

Examples

```
calls <- gt_batch(20)
calls$assumption
local <- gt_batch(20, by = "evaluator", sequential = 3, neighbor = TRUE)
c(sequential = local$sequential, neighbor = local$neighbor)

d <- expand.grid(item = 1:40, evaluator = 1:3, prompt = 1:2)
d$score <- sin(d$item) + d$evaluator / 5 + cos(d$item * d$prompt) / 4
design <- gt_design("item", c("evaluator", "prompt"), batch = calls)
report <- gt_preflight(d, "score", design)
report$batch_audit[c("membership", "batches", "conditions", "calls")]

# Calls recorded in the data are read, not reconstructed.
d$call <- paste(d$evaluator, d$prompt, (d$item - 1) %/% 20)
recorded <- gt_design("item", c("evaluator", "prompt"),
  batch = gt_batch(20, id = "call"))
gt_preflight(d, "score", recorded)$batch_audit[c("membership", "calls")]
```

gt_batch_dstudy *Compare Supplied Allocations for a Fixed Gaussian Batch Layout*

Description

Apply gt_batch_reliability to a bounded grid of instrumentation counts, retaining the same item grouping and the same explicitly weighted targets.

Usage

```
gt_batch_dstudy(fit, grid, contrasts = NULL, aggregate = NULL, score = NULL)
## S3 method for class 'gt_batch_dstudy'
print(x, ..., digits = 4L, max_rows = 12L)
## S3 method for class 'gt_batch_dstudy'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

fit	An accepted shared-call Gaussian fit with a recorded fixed layout.
grid	A nonempty data frame of named positive integer instrumentation counts, with at most 1000 rows. Unspecified facets retain fitted counts. Batch size and item count cannot be varied.
contrasts	Named item contrast weights, as in gt_batch_reliability.
aggregate	Named aggregate item weights, as in gt_batch_reliability.
score	Optional gt_score outcome weights.
x	A gt_batch_dstudy object.
digits	Significant digits for printing.
max_rows	Maximum number of printed rows.
row.names	Optional row names for tabular export.
optional	Accepted for generic compatibility; column names are preserved.
...	Reserved for additional print or data-frame arguments.

Details

Every candidate uses the target and assumptions documented in gt_batch_reliability. Name target weights using the item keys in gt_batch_reliability(fit)$layout$item. Numeric keys preserve the original values and can differ from ordinary printed numeric labels. All instrumentation facets remain random; this function cannot improve a projection by changing the universe to fixed facets. It neither generates allocations nor optimizes costs. It returns all supplied candidates, including repeated rows. No intervals are created. At most one million source-contribution rows are permitted, bounding the returned output size, checked before computing the first candidate. Annotations per item and total calls must remain below 2^{53} for exact integer accounting.

Value

A gt_batch_dstudy list with summary and source_contributions tables carrying a design_id referring to the corresponding row of grid. The grid includes all fitted instrumentation facets with unspecified counts filled in. The item layout, target weights, original counts, batch size, point-only uncertainty record and interpretation are retained. as.data.frame() returns the summary with allocation_<facet> columns joined by design_id, including unchanged fitted counts for unspecified facets. The prefix prevents facet names from colliding with summary or context columns. Additional columns retain pilot_items, batch_size, annotations_per_item, projected_calls, extrapolated, scale, uncertainty_available, uncertainty_reason, and interpretation through a CSV roundtrip. extrapolated identifies allocations with any instrumentation count larger than its fitted count. Grid, interpretation and uncertainty attributes are retained for compatibility. CSV does not preserve those attributes or the full item layout and target weights; retain the original list, for example with saveRDS(), when those details are needed to reproduce a weighted target.

See Also

gt_batch_reliability, gt_dstudy

Examples

```
candidates <- expand.grid(rater = 2:4, repetition = 1:3)
# gt_batch_dstudy(fit, candidates,
#   contrasts = list(pair = c(item1 = 1, item2 = -1)))
```

---

gt_batch_reliability *Project Gaussian Score Error Conditional on a Fixed Batch Layout*

Description

Compute annotation error for absolute item scores, named item contrasts, and weighted aggregates under an accepted shared-call Gaussian model. The calculation contracts source kernels directly without forming an observation covariance matrix.

Usage

```
gt_batch_reliability(fit, counts = NULL, contrasts = NULL,
  aggregate = NULL, score = NULL)
## S3 method for class 'gt_batch_reliability'
print(x, ..., digits = 4L, max_rows = 12L)
## S3 method for class 'gt_batch_reliability'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

fit	An accepted Gaussian gt_fit that models equal, fixed batches and records its item-to-batch layout. Older saved fits without this record must be refitted; membership is never guessed.
counts	Optional named positive integer instrumentation counts. Unspecified facets retain their fitted counts. Item count and batch size cannot be changed. All instrumentation facets remain random.
contrasts	A numeric vector named by item identifiers and summing to zero, or a named list of such vectors. Omitted items have weight zero. For example, list(within = c(item1 = 1, item2 = -1)) declares an item difference. Use the item keys in the returned layout$item; see Details.
aggregate	A numeric vector named by item identifiers, or a named list of such vectors. Every vector must have nonnegative weights summing to one. Omitted items have weight zero; weights are not normalized.
score	Optional gt_score weights for a composite across outcomes. Per-outcome results are also returned.
x	An object returned by this function.
digits	Significant digits for printing.
max_rows	Maximum number of rows printed.
row.names	Optional row names for the exported data frame.
optional	Accepted for generic compatibility; column names are preserved.
...	Reserved for additional print or data-frame arguments.

Details

Item keys. Call the function without target weights to obtain layout$item, then use these keys to name contrast and aggregate weights. Character identifiers and factor labels are unchanged. Plain integer and double identifiers use distinct character keys with 17 significant digits, independent of the digits, scipen, and OutDec options. Numeric keys always use a decimal point, including in comma-decimal locales. Reuse these keys as character strings; do not convert them back to numbers. These keys can differ from ordinary printed labels, particularly for large or fractional numbers. Positive and negative zero identify the same item. The fit retains the original typed identifiers for membership checks, including when observation retention is disabled. Row order does not define target identity.

Target and scope. The universe score is the object main effect. Each final item score averages a complete panel of all instrumentation conditions with equal weights. The fitted object grouping is held fixed. Facet counts may change under the model's exchangeability assumptions, but every planned condition must retain that same item grouping. The helper does not generalize over randomized regrouping or predict a change in shared-call variance when batch size changes. It does not support fixed instrumentation facets, unequal observation weights, incomplete panels, within-cell replication, or declared positional or facet-varying batch patterns that the fitted model did not estimate.

Error contractions. For a supplied item-weight vector w, a source containing the object contributes its variance times $\sum_i w_i^2$; an instrumentation-only source contributes its variance times $(\sum_i w_i)^2$.

Each is divided by the product of planned counts of the instrumentation facets in its grouping term. Residual variance is multiplied by $\sum_i w_i^2$ and divided by the number of annotations per item. The shared-call contribution is its variance times $\sum_b (\sum_{i \in b} w_i)^2$, divided by the number of annotations per item. For a multivariate score, each source variance is the corresponding quadratic form in `score`'s outcome weights.

Thus a shared call shift cancels from a same-batch item difference, remains in a between-batch difference, and remains in an absolute item score. A weighted aggregate depends on how its weights are distributed across batches. Instrumentation-only sources remain in an aggregate even as the number of averaged items increases. There is no universal batch penalty.

Interpretation of ratios. Each target has an explicit coefficient name:

- Absolute item scores: `Phi`.
- Item contrasts: `contrast_reliability`.
- Weighted aggregates: `aggregate_dependability`.

Each ratio divides the universe variance of its exact weighted target by that variance plus annotation-error variance. A contrast ratio is not a single G coefficient for every possible item ranking. Aggregate error conditions on the selected items and excludes corpus-sampling uncertainty. It is not uncertainty about an unknown population mean. Zero total variance returns `NA`.

All results are point projections; no sampling standard errors or confidence bounds are provided. The reported `error_sd` is the model-implied annotation-error standard deviation. It is not a standard error of an estimated coefficient.

At most one million source-contribution rows are permitted, checked before materializing target results. Nonfinite quadratic forms and numerically unrepresentable weight scales are refused; rescale extreme target weights.

The fitted layout and returned target weights contain item identifiers. Treat these objects as analysis data. Portable `gt_report(fit)` output does not include the layout; this result is not silently inserted into it.

Value

A `gt_batch_reliability` list. `summary` has one row per target and outcome, plus requested composites, with universe variance, error variance, error standard deviation, ratio and its explicit name. `source_contributions` records each source's role, kernel contraction, divisor and variance contribution. `layout` and `target_weights` record the item keys, grouping and weighted targets. Other fields record fitted and planned facet counts, pilot-item count, annotations per item, projected calls, unchanged batch size, extrapolation, scale and interpretation. `uncertainty$available` is `FALSE`. `as.data.frame()` returns `summary` and retains counts, interpretation and uncertainty as attributes; CSV output does not retain those attributes.

See Also

`gt_batch`, `gt_batch_dstudy`, `gt_reliability`, `gt_score`

Examples

```
# Weights identify the target explicitly; use fitted item IDs in an analysis.
within_pair <- c(item1 = 1, item2 = -1)
```

```
corpus_weights <- c(item1 = 0.25, item2 = 0.25, item3 = 0.5)
# gt_batch_reliability(fit, contrasts = list(within = within_pair),
#   aggregate = list(corpus = corpus_weights))
```

---

gt_bootstrap *Object Bootstrap of an Accepted Fit*

Description

Resamples the objects of measurement with replacement, each with its complete instrumentation panel, refits every resample under the fit's model and the recorded refit controls, and keeps the accepted replicates' source covariance matrices. gt_reliability and gt_dstudy then take percentile or BCa intervals from them, and summary() gives intervals for the variance components.

Usage

```
gt_bootstrap(fit, replicates = 1000L, seed, max_attempts = 2L * replicates,
  jackknife = FALSE, warm_start = TRUE, cores = 1L, data = NULL,
  control = NULL)
## S3 method for class 'gt_bootstrap'
print(x, ...)
## S3 method for class 'gt_bootstrap'
summary(object, level = 0.95,
  interval_type = c("percentile", "bca"), ...)
## S3 method for class 'gt_bootstrap'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

fit	A Gaussian or discrete fit that was numerically accepted, of a panel in which every object shares one instrumentation layout. A fit with declared batches is refused, because resampling items would break its fixed call layout, and so is a fit whose objects do not share one layout, because a resample could lose a facet level confined to the objects it omits.
replicates	The number of accepted refits wanted.
seed	Required integer seed between zero and .Machine$integer.max. The caller's random-number state is restored.
max_attempts	The largest number of resamples to try; it cannot be smaller than the number of replicates wanted. Resampling stops when that many refits are accepted or the budget is spent.
jackknife	TRUE also refits the data with each object left out once, which BCa intervals need for their acceleration. This costs one refit per object.
warm_start	For discrete fits, TRUE starts each refit's primary attempt at the original estimates. A source on its covariance boundary starts from the original fit's starting values instead, so no refit begins on a zero factor column. The alternative starts and every acceptance check still apply. Gaussian refits always start as the original fit did.

`cores` Processes to use. More than one forks the R session through the parallel package, which cannot fork on Windows. Each attempt draws from its own seed, so the replicates do not depend on this value.

`data` The original modelled observations and design identifiers, in their fitted row order; needed when the fit was stored without its data. A supplied copy is checked against retained observations or a compact identity recorded before data retention. Matching only the panel dimensions is insufficient. Row names, unused columns, column order and factor storage do not matter; numeric identifiers are compared by exact value, treating integer and double storage as equivalent. Numeric identifiers are distinct from character labels; factor and character labels are equivalent. Number-display options such as `scipen` and `OutDec` do not affect identity verification. Permuting rows is refused because object encounter order determines seeded resampling. Original category labels and the fit's retained binary codes are accepted under the fit's recorded category order. As in `gt_fit()`, original numeric or logical binary observations use labels `c("0", "1")`; reversed or other labels require a factor. Numeric 0/1 supplied for a fit with other labels is interpreted as its retained internal codes. Older fits that kept neither observations nor a verifiable identity must be refitted. Identity format 2 preserves exact numeric identifiers. Older format-1 identities are not accepted without retained observations: their lost numeric precision cannot be recovered. Fits with retained observations can derive format 2 directly. Bootstrap objects carrying format-1 identities must be regenerated before attaching intervals.

`control` A `gt_control` for the refits; by default the fit's own. Explicit controls set the refits' numerical settings and acceptance thresholds, so they need not match those used for the original fit. They must preserve any `fixed_covariance` matrices: adding, removing or changing fixed source covariances would change the model and is refused. Equivalent source and matrix-label ordering is allowed. The bootstrap does not automatically relax acceptance thresholds. Refits compute no observed information, which plays no part in acceptance.

`x`, `object` A `gt_bootstrap` object.

`level` Confidence level for the component intervals.

`interval_type` `"percentile"` or `"bca"`; BCa needs `jackknife = TRUE`.

`row.names`, `optional`, `...`
Generic arguments; `row.names` sets the record's row names, and the others are unused.

Details

The resample. Objects are drawn with replacement, as many as were fitted. A drawn object keeps all of its rows, so its complete panel of evaluators, prompts and other facet levels travels with it, and an object drawn more than once enters the refit as that many distinct objects. Every object must share one instrumentation layout: the same facet-level combinations, as often, with the same outcomes observed. The facet levels are then the observed ones in every replicate, and objects that share one partial layout keep it. When the layouts differ, a level confined to some objects would be lost from every resample that omits them, so such a fit is refused. Resampling the object facet is the boot-p procedure of Brennan (2001). No resampling scheme reproduces the sampling of a crossed random design exactly (McCullagh, 2000).

What the intervals describe. Object-sampling uncertainty, conditional on the observed instrumentation levels, on the declared model, and on numerical acceptance. They do not reflect that the evaluators, prompts or other facet levels were themselves a sample, which the random-facet Wald intervals do; sources without the object, such as an evaluator main effect, are therefore estimated from the same levels in every replicate. Relative coefficients depend mainly on object-level sources; absolute coefficients also depend on facet main effects, so their bootstrap intervals can be narrower than the corresponding Wald intervals. Neither is a calibration guarantee.

Acceptance. Each refit uses the fit's design, families, estimator and resolved covariance structures, including fixed source covariances. Numerical checks use the recorded refit controls, which default to the original controls and may be explicitly overridden. A refused or failed refit is recorded with its reason and never stands in for an accepted one. Resampling continues to `max_attempts`; when fewer than `replicates` are accepted the object says so. Dropping refused refits conditions the intervals on fitting success; report the attempt record beside them.

Intervals. The original fit supplies every point estimate. Percentile intervals are type 6 sample quantiles (see `quantile`) of the finite replicate values. BCa intervals adjust the percentiles for the share of replicates below the estimate and for the leave-one-out skewness (Efron, 1987). An interval needs a finite original estimate and at least 50 finite replicate values; BCa also needs every leave-one-out refit accepted and finite. An endpoint whose percentile falls outside $[1/(B+1), B/(B+1)]$, for B finite values, has no replicate support and is reported as `NA`. The bootstrap standard error is the standard deviation of the replicate values.

Coverage. In a frozen pilot of the Gaussian item-by-rater design with 24 items and three or six raters, 200 REML panels per setting and 500 replicates each, 95% percentile intervals covered the population coefficients in 92.0–94.0% of panels for Erho2 and 89.5–92.5% for Phi; BCa intervals were available in 85–99% of panels and covered in 89.1–93.8% of those; delta-method intervals on the same panels covered in 95.0–97.0%. The shortfall also occurred at zero rater variance. Coverage for other designs, larger panels and discrete or multivariate fits has not been assessed. The protocol and results are in the package repository under '`validation-studies/bootstrap-coverage`'.

Cost. A bootstrap costs one fit per attempt, and the leave-one-out refits one fit per object. Small Gaussian fits take a fraction of a second each; joint categorical fits of large panels can take an hour or more each, which makes a bootstrap of them a matter of days even across many cores. For those, the boundary-conditional Wald intervals of `gt_control(discrete = list(check_hessian = TRUE))` remain available.

Value

A `gt_bootstrap` list with the resampled unit (the object column), the number of objects, the requested and accepted replicate counts, the number of attempts, the accepted replicates' source covariance matrices in attempt order, an attempt record, the leave-one-out refits when requested, the original fit's estimates that identify it, the settings, and an interpretation.

The attempt record, also returned by `as.data.frame()`, has one row per attempt up to the last accepted replicate used: the attempt number, its resampling seed, the retry seed the refit used (an explicit Gaussian `retry_seed` in the controls, which every refit then uses, or else one from the same stream), its status (`"accepted"`, `"refused"` or `"error"`), the acceptance failures or error message, elapsed seconds, the number of warnings, the deviance, and the sources on their covariance boundary. The leave-one-out record names the object left out instead of a resampling seed.

The settings retain `jackknife`, the requested `gt_control`, and a `refit_control` template after the bootstrap's information and warm-start adjustments. Per-attempt retry seeds come from the records

unless an explicit Gaussian retry seed is present. The template does not reconstruct engine defaults or optimizer-internal restart states; retain the source archive and numerical environment alongside it.

New bootstrap objects also record the fitted data identity, which is checked when attaching intervals to a fit or report. Legacy bootstrap objects without this identity retain the older estimate-based association checks; they cannot verify that the underlying observations and identifiers match.

`summary()` returns a data frame with one row per unique entry of each source covariance matrix: the original estimate, the bootstrap standard error, the interval bounds, and the number of finite replicate values.

References

Brennan, R. L. (2001). *Generalizability Theory*. Springer. doi:10.1007/9781475734560

Efron, B. (1987). Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397), 171–185. doi:10.1080/01621459.1987.10478410

McCullagh, P. (2000). Resampling and exchangeable arrays. *Bernoulli*, 6(2), 285–301. doi:10.2307/3318577

See Also

`gt_reliability`, `gt_dstudy`, `gt_fit`

Examples

```
set.seed(2)
d <- expand.grid(rater = factor(1:4), item = factor(1:30))
d$score <- rnorm(30)[d$item] + rnorm(4, sd = 0.5)[d$rater] + rnorm(nrow(d), sd = 0.7)
fit <- gt_fit(d, "score", gt_design("item", "rater"))

boot <- gt_bootstrap(fit, replicates = 200, seed = 11)
boot
gt_reliability(fit, bootstrap = boot)
gt_dstudy(fit, data.frame(rater = c(1, 2, 8)), bootstrap = boot)
summary(boot)
```

---

gt_categorical_target *Integrate Explicit Repeated-Panel Categorical Targets*

Description

Projects observed-score, category-vector or plurality-decision targets by drawing independent pairs of complete future panels from an accepted categorical fit. The fit is held fixed.

Usage

```
gt_categorical_target(fit, target = c("score", "category_vector", "decision"),
  counts = NULL, scores = NULL, n_items = 200L, replications = 40L,
  seed, max_rows = 100000L, outcome = NULL)
## S3 method for class 'gt_categorical_target'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
## S3 method for class 'gt_categorical_target'
print(x, ..., digits = 4L)
```

Arguments

fit	A numerically accepted binary, ordinal, nominal or joint categorical gt_fit. Mixed Gaussian/categorical fits are not supported.
target	"score" for the mean of explicitly scored responses, "category_vector" for the vector of empirical category proportions, or "decision" for repeated-panel plurality agreement.
counts	Named future facet counts, each at least one; omitted facets retain their fitted counts. All instrumentation facets are random.
scores	Named numeric category scores for a single selected outcome, or a named list of these vectors by outcome. Binary outcomes default to 0 and 1 in the declared level order. Ordinal outcomes require explicit, strictly increasing scores. Nominal outcomes use the vector target.
n_items	Objects drawn per integration panel, at least two. This controls integration precision, not the amount of pilot information. For modelled calls it must equal the fitted object count, retaining its fixed equal call membership.
replications	Independent panel pairs, from 2 to 10000. Two permits computation but does not imply adequate integration precision.
seed	Required nonnegative integer seed. The caller's RNG is restored.
max_rows	Maximum rows per simulated panel, no greater than one million. The function also limits all generated rows to ten million and estimated workspace to 512 MiB.
outcome	Exact outcome names to report; NULL selects all. All fitted dimensions are simulated so their joint law is preserved.
x	A gt_categorical_target object.
row.names	Optional data-frame row names.
optional	Accepted for generic compatibility.
digits	Significant digits for printing.
...	Reserved for generic compatibility.

Details

Each panel pair shares only the object's main latent random effect. Instrumentation, object-by-instrumentation interactions, and modelled calls are independently redrawn between panels. Dependence within a panel and between outcomes is retained. This defines a population of exchangeable

objects and instrumentation levels under the fitted model; it does not condition on fitted random-effect modes or on named observed raters.

For a scalar panel score S, the target is

$$\mathrm{Var}\{\mathrm{E}(S \mid U)\}/\mathrm{Var}(S),$$

where U is the stable object effect. Equivalently the numerator is the covariance between independent repeat scores given U. Random instrumentation shifts enter total variance. This is an absolute repeated-panel score ratio, not the relative latent G coefficient. Conditional response means and second moments are integrated analytically given the sampled effects to reduce Monte Carlo noise.

For category-proportion vector P, replace each variance by the trace of its covariance matrix. This ratio is invariant to a permutation of category coordinates and assigns no artificial numeric distances to unordered labels. It is not Krippendorff's alpha or a chance-corrected agreement coefficient. Positive affine transformations of scalar scores leave the score ratio unchanged.

The decision target is the probability that the plurality category agrees between panels. Ties are broken uniformly and independently in each panel; the tie randomness is integrated exactly. It reports the marginal decision probabilities and `decision_diversity`, one minus their squared sum. A constant decision can have agreement one and diversity zero. Agreement is not discrimination, correctness, or validity against human judgments. Other decision rules and fixed-facet targets are not implemented.

Monte Carlo standard errors use independent entire panel pairs as the replication units, because items within one panel share instrumentation. Ratio standard errors use a delta method on these replicate moments. They quantify numerical integration error at the fitted parameter values, not estimation uncertainty, coverage, or future-panel prediction. No parameter confidence interval is returned. Ratios with unresolved denominators or estimates outside [0,1] remain unavailable; their raw estimates and reasons are retained. Increase integration precision and inspect the model rather than treating a refused estimate as zero.

With weak positive object signal, a finite integration budget can return many unavailable ratios. Summarizing only the available ratios can then select upward-biased estimates, even when their conditional error bands appear adequate. Retain every attempted seed and its availability; do not replace refused runs or clip their raw estimates into [0,1]. More objects within a panel cannot remove Monte Carlo variation from shared instrumentation. Independent panel pairs are needed. As a pilot approximation, multiplying the pair count by the squared ratio of the current to the desired Monte Carlo standard error suggests a new budget, within the resource limits. Verify that budget with independent seeds and report availability as well as precision; this scaling rule is not a guarantee near zero signal or an unresolved denominator.

Future panels are complete. Using a fit from missing whole cells requires the same exchangeability and ignorable observation assumptions as that fit; the simulation does not impute historical labels or establish ignorability. Modelled calls retain the fitted equal object membership and redraw call effects at each future facet combination. Variable batches and sequential carry-over effects are refused.

Value

A `gt_categorical_target` object. The data-frame method exports these fields:

`outcome`, `target` Outcome and estimand identifiers.

`estimate`, `raw_estimate` Accepted and raw numerical estimates.

`mc_se` Whole-panel integration standard error.

`signal_variance` Stable-object variance, absent for decisions.

`total_variance` Total score variance, absent for decisions.

`available`, `reason` Availability and any refusal reason.

`decision_diversity` Diversity of the marginal plurality decision.

Exports also retain integration counts, batch status and a false `parameter_uncertainty` flag.

Other list fields are:

- `allocation`: projected object and facet counts.
- `scores`: supplied category scores.
- `decision_probabilities`: the marginal decision law.
- `integration`: simulation settings.
- `shared_sources`: effects retained between panels.
- `redrawn_sources`: effects redrawn between panels.
- `decision_rule`: the specified vote rule.
- `interpretation`: the estimand definition.
- `integration_moments`: the whole-panel moments.

For numerical stability, scalar-score moments use scores normalized to [0,1]. The reported signal and total variances use the original score units.

The `integration` metadata contain these additional entries:

- `score_transform`: the offset and span.
- `resource_budget`: estimated moment storage and peak working storage, including one active panel pair and target calculation scratch.

The combined estimate must not exceed 512 MiB. This is a conservative allocation guard, not a bound on the entire R process's memory use.

See Also

`gt_simulate`, `gt_reliability`, `gt_plan`

Examples

```
d <- expand.grid(item = 1:8, rater = 1:3)
d$label <- rep(c(0, 1), length.out = nrow(d))
fit <- gt_fit(d, label ~ item + rater, measured = "item",
  family = gt_family("binary"),
  control = gt_control(discrete = list(optimizer = "nlminb",
    fixed_covariance = list(item = matrix(0, 1, 1),
      rater = matrix(0, 1, 1)))))
result <- gt_categorical_target(fit, "score", n_items = 20,
  replications = 5, seed = 91)
as.data.frame(result)
```

gt_components *Extract Source Covariances or Correlations*

Description

Returns source-specific matrices on their modeled scale without transforming latent variances into observed-score variances.

Usage

```
gt_components(fit, correlation = FALSE, tolerance = 1e-8)
```

Arguments

fit	A gt_fit object.
correlation	Whether to return correlation matrices instead of covariances.
tolerance	Nonnegative relative variance threshold for reporting correlations. Entries involving a variance at or below this threshold times the reference variance scale are NA.

Details

Gaussian correlation checks use observed outcome variances as the reference scales. When those are unavailable, the largest variance within each source matrix is the reference. Covariances for discrete outcomes refer to identified latent predictors or category contrasts. Correlations involving negligible source variances are suppressed rather than presented as stable estimates.

Value

A named list of matrices, with scale and interpretation attributes. Correlation extraction adds the threshold and boundary policy.

See Also

gt_fit, gt_diagnostics

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
gt_components(fit)
```

gt_control *Configure Gaussian and Discrete Fitting and Acceptance Checks*

Description

Collects backend controls in separate named lists, together with which optional components a fitted object keeps. Backend-specific names and values are checked when a model is fitted.

Usage

```
gt_control(gaussian = list(), discrete = list(), retain = list())
## S3 method for class 'gt_control'
print(x, ...)
```

Arguments

gaussian	Uniquely named list of Gaussian settings, described below.
discrete	Uniquely named list of discrete settings, described below.
retain	Uniquely named list of logical retention switches, described below. Every default is TRUE.
x	A gt_control object for the print method.
...	Further print arguments; currently unused.

Details

Retained components. retain decides which optional parts of a fitted object are kept. All four default to TRUE; dropping a component is always something the caller asked for.

- data: the modelled data frame. An lme4 model cannot be separated from its data and is dropped with it. Dropping it removes every copy of the observations from the fit, including the summary copy the Gaussian OpenMx model carried and the data argument of the recorded call, which is replaced by a marker. Compact item-to-batch design labels remain on a supported shared-call fit so that gt_batch_reliability() can evaluate named targets; this switch does not de-identify the fitted design. Portable reports exclude these labels.
- model: the Gaussian OpenMx model and its factorial cross-products, and for a discrete fit the conditional predictors and probabilities held for every observation. The OpenMx model carries its own copy of the observations until data is dropped, so with them this is usually the largest component. The observed outcome variances are kept either way, so the correlations that gt_components() can report are unaffected.
- retry_log: the optimizer transcript and the bulky per-attempt payloads. Each attempt's label, error, optimizer code and message, and objective are kept, so gt_diagnostics() reports the same decisions and reasons.
- session: the sessionInfo() record. Keep it for reproducible research; it is worth its size when a result must be traced to an environment.

None of these components is used to compute an estimate, a diagnostic decision, a coefficient, or a decision study. `gt_reliability()` and `gt_dstudy()` work from the fitted source covariances, the resolved design, and either the retained data or the compact panel summary every fit records. With `data = TRUE` the balanced-panel rules are checked against the retained data, so a panel edited after fitting is still caught; with `data = FALSE` the same two rules are applied to the recorded summary. The fit keeps a `retained` record of what was dropped.

Gaussian controls and their defaults are:

- `engine`: `"openmx"`, the exact balanced likelihood optimized through OpenMx. `"lme4"` fits the same random sources as random intercepts with `lme4::lmer()`, one outcome at a time, with the call source when a balanced batch is modelled. An lme4 fit is accepted only when lme4's optimizer completes without any lme4 convergence warning; when the package's own stationarity check on lme4's deviance function passes, which computes lme4's Cholesky-scaled gradient statistic over the interior parameters against lme4's tolerance and requires that no parameter resting on its zero bound has a one-sided derivative pointing inward (a source variance of zero is a legitimate boundary estimate, so lme4's singular-fit message is suppressed and the boundary is judged by that derivative instead); when a restart from a different point itself completes without a convergence warning and reproduces objective and parameters; and when an independently calculated Gaussian deviance agrees with lme4's criterion. The reference is the factorial-contrast likelihood for `panel = "complete"` and the dense observed-row likelihood for `panel = "observed"`. This agreement verifies the model and objective, rather than establishing optimality. A verdict the statistic cannot support is a refusal, never an acceptance. It computes no standard errors, so delta-method coefficient intervals are unavailable; for a complete panel without declared batches, `gt_bootstrap` supplies object-bootstrap coefficient intervals. The exact engine's settings `optimizer`, `max_iterations`, `tolerance`, `check_hessian`, `threads`, `extra_tries`, `start` and `initialization` are refused with it. lme4 is a suggested package.
- `panel`: `"complete"`. The explicit `"observed"` option requires `engine = "lme4"` and fits a univariate Gaussian outcome to observed rows when whole cells are absent. It does not fill cells or remove missing outcome values, and refuses within-cell replication, joint outcomes and batch declarations. Source and residual covariance kernels must remain linearly independent after removing the intercept. See gaussian_observed for the assumptions and distinction between fitting an incomplete panel and projecting to an intended balanced allocation.
- `max_observed_rows`: 1000, used only with `panel = "observed"`. This explicit bound limits the dense independent likelihood check. Raising it does not qualify larger designs statistically; the memory guard also applies.
- `optimizer`: CSOLNP.
- `max_iterations`: 3000.
- `tolerance`: 1e-12.
- `check_hessian`: TRUE.
- `threads`: 1.
- `silent`: TRUE.
- `extra_tries`: 9.
- `retry_seed`: NULL.
- `start`: NULL (automatic moment-based covariance starts).

- `initialization`: `"automatic"`; `"none"` disables the square-root recovery described below.
- `max_preparation_bytes`: 512 * 1024^2.

Square-root recovery. With `initialization = "automatic"`, a balanced multivariate model with at least one unstructured covariance and no call effect receives one deterministic recovery after an ordinary optimizer attempt is refused while an attempt remains in the `extra_tries` budget. A bounded L-BFGS-B search (at most 10,000 iterations, `factr = 1000`, `pgtol = 0`, a fixed metric of standard-deviation units) minimizes the square-root form of the same exact deviance: each stratum covariance root is the QR factor of the stacked weighted source factors and the cross-products enter through their own roots, so nearly singular strata are evaluated without forming and inverting their covariance. The search starts from the model's declared original coordinates, never from the refused endpoint. Its endpoint's covariance components become the starting components of the next ordinary attempt, through the same path as a user-supplied `start` so that the parameter scales are recomputed for them, only if the search reports convergence, the point lies within the bounds, and the independent deviance evaluation reproduces the square-root value within 1e-5; otherwise the ordinary perturbed restarts continue from the best attempt whose likelihood was independently reconstructed, or from the original start when there is none. The recovery consumes an existing attempt, adds none, and decides nothing: acceptance still requires optimizer completion, the covariance-scale stationarity check and the independent likelihood check, with unchanged tolerances. `gt_diagnostics(fit)$initialization` records the policy and a status of `"disabled"`, `"not_applicable"`, `"not_invoked"`, `"handoff_eligible"` or `"refused"`, with the trigger and handoff attempts, the initializer's optimizer code, message and evaluation counts, and the independent deviance difference. An accepted primary attempt is identical under both settings. The control is refused with the lme4 engine.

The allocation guard estimates preparation memory. It excludes caller data and downstream engine objects. For observed-row panels, it includes dense covariance kernels, working matrices and the identification calculation after removing the intercept.

An explicit retry seed preserves the caller's random-number state.

Reproducibility. Retain the exact package archive and checksum, data and row order, response coding, source specification, all controls and starting values, and the R, dependency and numerical-library versions. For Gaussian restarts, set `retry_seed` explicitly and keep the thread count fixed. Record separate seeds for simulation and Monte Carlo diagnostics. A seed fixes the random draws it controls; it does not fix platform-dependent floating-point arithmetic or guarantee bitwise-identical optimized estimates.

Replay a request in the same recorded environment before comparing it across platforms. Compare matched unrounded quantities under numerical tolerances declared before the rerun, together with numerical-acceptance and uncertainty-availability decisions. A different decision is a reproduction failure to investigate, even if rounded coefficients look similar. Preserve that result instead of increasing a tolerance after seeing it. The installed '`examples/native_1_0_workflow.R`' records a comparison policy for its two specified native models; those tolerances are not general optimizer settings or a guarantee for other studies. See gt_fit for the distinction between numerical agreement, approximation adequacy and statistical calibration.

Gaussian `start` accepts a complete named list of covariance matrices for all model sources, including `Residual`. Fully named matrix axes are matched independently to the outcome names; both unnamed axes use outcome order. Incomplete, duplicate, or incompatible labels are rejected. Admissible matrices receive the engine's existing interior eigenvalue floor. Diagonal models retain

their repaired diagonal entries, and a pooled residual starts from the mean of those entries. These values initialize estimated parameters; they do not fix their fitted values.

Discrete resource limits are:

- `engine`: `"dense"`; `"sparse"` selects sparse random-design, conditional-curvature and factorization calculations for binary, ordinal and nominal outcomes, separately or jointly. Both calculate the same first-order Laplace objective and apply the same final numerical-acceptance requirements; their conditional-mode search paths can differ. This switch does not establish full-panel feasibility or approximation accuracy.
- `max_random_dimension`: 200.
- `max_observations`: 1200.
- `max_parameters`: 80.
- `max_dense_bytes`: 512 * 1024^2.
- `max_sparse_bytes`: 512 * 1024^2.

Only the selected engine's byte limit is applied. Observation, random-dimension and parameter limits apply to both engines. The sparse estimate is made in two stages. A complete-fill bound, which allows fully populated source covariance factors and complete fill of the conditional Hessian and its factor, is kept whenever it already fits the budget. Otherwise the exact count of the conditional Hessian's structural nonzeros (two coordinates are coupled only when their groups share an observation, so the count follows from the observed pairs of group levels) decides whether the Hessian alone exceeds the budget, in which case the request is refused before any pattern is built; if it does not, the structural pattern with every factor entry nonzero and every latent dimension coupled is factorized once with a fill-reducing ordering computed for that pattern, and the counted Hessian and factor entries give the estimate. The method and the counts are recorded by `gt_preflight()`. It remains a planning estimate rather than a bound: the fit recomputes its ordering on the fitted matrix, whose pattern is contained in the structural one, so a fitted factor can fill slightly more or less than the structural factor did, and the multiplier is the allowance. The estimate excludes caller data, optimizer state and the repeated allocations of the finite-difference checks, and a feasible factorization does not establish acceptance or approximation accuracy at that size. Discrete optimizer controls are:

- `maxit`: 150.
- `inner_maxit`: 60.
- `inner_tol`: 1e-7.
- `reltol`: 1e-7.
- `start_sd`: 0.4.
- `trace`: 0.
- `optimizer`: `"L-BFGS-B"`; alternatively `"nlminb"`. A coarse L-BFGS-B attempt that stops at its start after an invalid evaluation is rerun at a parameter scale of one over the square root of the number of rows, so its first step is the per-row gradient.
- `optimizer_scaling`: `"none"`; optionally `"start"` for L-BFGS-B. Each attempt then uses a fixed coordinate scale `pmax(1, abs(start)) / sqrt(nrow(data))`, computed from that attempt's starting vector and recorded with its optimizer controls. This changes the search path, not the likelihood, finite-difference steps in original coordinates, or independent acceptance

checks. The projected-gradient stopping tolerance `pgtol` is applied to the scaled gradient and is not rescaled, so a scaled attempt can stop earlier in original-gradient terms; the independent stationarity and restart checks are unchanged. The automatic collapsed-step rescue does not replace an explicitly requested metric. Scaling is not supported with `"nlminb"`.

- `gradient`: `"finite_difference"`; optionally `"analytic"` with `engine = "sparse"` and L-BFGS-B. Every optimizer attempt then receives the analytic gradient of the first-order Laplace objective, from a selected inverse of the conditional Hessian's factor in compiled code, in place of finite differences; the gradient reuses the objective evaluation at the same point. A gradient that cannot be computed at a usable point ends its attempt as an optimizer error; there is no fallback inside an attempt. Acceptance is unchanged: the tight rerun, alternative starts, the stationarity check with its own finite differences, restart stability and the bound rule decide as before. The route supports natural Cholesky and log-Cholesky source covariance coordinates and fixed covariances; direct and exchangeable variance coordinates are refused at fit start, because their derivative at an exact zero variance is not available, so a univariate or diagonal model needs `covariance_parameterization = "cholesky"`. The design must also lie within the compiled routines' declared caps, which `gt_preflight()` reports.
- `boundary_escape`: `"none"`; optionally `"projected"` for natural Cholesky source covariance coordinates. After each completed tight attempt, every source whose covariance is singular is checked in covariance coordinates, as the stationarity check does. A source whose projected score points into the interior, beyond `stationarity_tol`, has its covariance moved by a projected step that keeps it positive semidefinite, with the step length chosen by the objective, and the tight attempt is rerun from there, at most three times. In natural Cholesky coordinates the derivative along an exact zero factor column vanishes, so the analytic route's L-BFGS-B attempts cannot leave such a boundary by their gradient alone; the finite-difference route's one-sided difference at the bound can. Each rerun is an attempt of its own in the record, and `fit$diagnostics$boundary_escape` keeps every step's singular sources, scores, step length, objective change and evaluation count. A rerun that raises an error, returns no usable result or does not lower the tight objective ends the escape, and the earlier result stands. A rerun that lowers the objective but stops at the iteration limit becomes the candidate and also ends the escape; acceptance then refuses it as incomplete. `optimization_trial_budget` in the diagnostics counts the three possible reruns of every tight attempt. Acceptance is unchanged, and a source legitimately at zero, whose score does not point inward, is left alone. Other coordinates are refused at fit start.
- `start`: NULL (automatic response and covariance starts).
- `parameter_bounds`: NULL (the default numerical domains).
- `covariance_parameterization`: `"auto"`.

The default `"auto"` uses direct nonnegative variance coordinates for univariate models and for joint models whose `covariance` is `"diagonal"`. Joint models with unstructured covariance use `"cholesky"`: the entries of each source's lower-triangular factor, whose nonnegative diagonal can reach zero, so a singular source matrix is represented exactly. All cross-outcome covariance parameters are retained. Explicit `"variance"`, `"cholesky"` and `"log_cholesky"` selections remain available; log-Cholesky coordinates can only approach a singular matrix. Direct variance coordinates admit exact zeros; an explicit `"variance"` request with joint unstructured covariance is rejected rather than removing covariances. The default variance upper bound is `exp(10)`, the square of the default maximum diagonal factor entry, `exp(5)`, in Cholesky and log-Cholesky coordinates. Off-diagonal factor entries default to bounds of -10 and 10. The exchangeable covariance for nom-

inal outcomes always uses direct variance coordinates; the Cholesky choices are refused with it. `start_sd` still denotes a starting standard deviation and must be positive.

Discrete `start` is a finite, uniquely named numeric vector of optimizer parameters, with complete or partial overrides of the automatic starts. Names are matched exactly; unknown names and values outside the model bounds are rejected. Automatic starts are also checked against the bounds before fitting; the optimizer never silently projects an invalid starting vector. Labels that produce ambiguous parameter names are rejected. The fitted `fit$parameters` vector can initialize a refit of the same model, with unchanged family links, category coding, covariance structure, and covariance parameterization. These are stored parameter coordinates, before optimizer scaling: ordinal threshold parameters contain the first threshold followed by log positive gaps; Cholesky parameters are factor entries; log-Cholesky diagonal parameters are log standard deviations; direct-variance parameters are variances. Actual and automatic initial vectors are retained in the diagnostics, and `fit$starting_parameters` records the primary start. Alternative starts perturb fixed-effect coordinates from the primary start and use the automatic covariance starts with distinct deterministic scale perturbations. If clipping would duplicate an earlier start, a bounded fixed-effect perturbation makes the additional trial distinct.

Discrete `parameter_bounds` is an optional list with named numeric `lower` and/or `upper` vectors. These partially override artificial optimizer bounds in the same resolved coordinates used by `start`; unknown or duplicate names, nonfinite values, and unordered bounds are rejected. For example, `list(lower = c("rating::threshold_parameter1" = -30))` extends the first threshold's lower bound for outcome `rating`. Natural variance and ordinary Cholesky diagonal lower bounds must remain zero. Declared domains must keep ordinal gaps positive and distinguishable and covariance factors and summed variances finite. Starts must lie in the effective domain. Defaults are unchanged, and bounds never expand automatically. A wider numerical domain changes neither the likelihood nor the acceptance thresholds: contact with a declared artificial bound still refuses acceptance. It does not establish identification, approximation accuracy or interval coverage. Preflight validates these declarations after preceding structural, category and resource checks pass. Diagnostics retain a `parameter_domain` list. Its `requested`, `default` and `effective` entries describe the declared and resolved lower/upper vectors. Its `natural_zero_coordinates` entry names the natural boundaries. The separate `parameter_bounds` field continues to list contacted bounds. Retained controls and Laplace diagnostics must reproduce the same domain. Declare any domain revision before a new run and preserve the earlier refusal as evidence.

The selected discrete optimizer remains fixed for primary fits, alternatives, and tight refinements; the engine never silently switches algorithms. `maxit` limits iterations per optimization. For `nlminb`, the function evaluation budget is `max(200, 10 * maxit)`, and `reltol` supplies its relative objective tolerance. Each optimizer uses its own native stopping rules; the same independent numerical-acceptance checks apply afterward. Optimizer choice is not evidence of approximation accuracy.

Zero is a natural lower boundary in direct variance coordinates and on ordinary Cholesky factor diagonals. Within `gt_diagnostics(fit)$diagnostics`, `zero_variance_parameters` retains the names of exact-zero natural-boundary coordinates. For variance or exchangeable coordinates these are variance parameters. For Cholesky coordinates they are zero diagonal factor pivots; the corresponding marginal variance can still be positive. Use `gt_components` for the source covariance entries, and `boundary_sources` for the recorded zero or nearly singular sources. These estimates must still pass independent covariance-scale stationarity and restart checks. All upper bounds and fixed-effect/threshold bounds remain artificial and still cause rejection on contact. This mode does not establish approximation accuracy or sampling properties for boundary estimates. The diagnostics field `covariance_parameterization_requested` records the con-

trol selection; covariance_parameterization records the resolved coordinates ("variance", "cholesky", "log_cholesky", "exchangeable" or "fixed").

Discrete numerical-acceptance controls are:

- stationarity_tol: 1e-3.
- validation_reltol: 1e-10.
- validation_inner_tol: 1e-9.
- alternative_starts: 1.
- stability_objective_tol: 1e-6.
- stability_parameter_tol: 0.02.
- bound_tol: 1e-4.

Restart stability compares all tight candidates. The objective difference is the absolute difference from the selected objective divided by max(1, abs(selected objective)). The parameter distance is the maximum of two quantities: the largest absolute difference in stored intercept/threshold coordinates, each divided by max(1, abs(left), abs(right)); and, for each source, the largest absolute entry difference between its two covariance matrices, divided by the maximum of one and the largest absolute entry in either matrix. Covariance matrices are compared after reconstructing them from their factors, so equivalent Cholesky factors have distance zero. Every tight candidate must complete and satisfy both tolerances; the remaining numerical-acceptance requirements still apply.

Setting alternative starts to zero disables numerical acceptance, not just an optional report. The discrete optimization-attempt budget is 2 * (1 + alternative_starts): one preliminary and one tight fit for the primary start and each alternative. Final-mode and stationarity evaluations are recorded separately and are not additional optimization trials. Gaussian extra_tries and discrete alternative_starts therefore have different meanings. Covariance derivative accuracy uses at most six step halvings when needed, without changing the stationarity or derivative-disagreement tolerances. Every stationarity evaluation, including the center and all fixed-effect and covariance perturbations, must have a valid, converged conditional mode and usable finite likelihood. A failed evaluation ends the diagnostic and records a validation error; an optimizer penalty is never treated as a derivative.

For a known-covariance calculation, fixed_covariance can provide one positive-semidefinite matrix for every discrete random source. Fully named matrices are matched and reordered to the latent dimensions; unnamed matrices are positional. Partial or incompatible dimension names are rejected. These matrices are fixed assumptions rather than estimated source covariances. The covariance parameterization setting is not used when all covariances are fixed.

The attempts entry of gt_diagnostics(fit)$diagnostics retains starts, controls, optimizer codes, messages, errors, warnings, elapsed time, and returned parameters. A computational failure during refinement, a restart, or stationarity checking prevents numerical acceptance but preserves an otherwise usable primary or alternative fit for inspection. Unusable optimizer returns are retained unchanged as raw_result, separately from the computational placeholder. The selected attempt, optimizer, trial count, and budget are also reported. If no fitted candidate supplies a usable conditional mode, a gt_discrete_numerical_failure error condition retains the attempt records instead. Invalid input still fails before fitting. Warnings captured during numerical attempts are retained in these diagnostics; user interrupts are not swallowed.

For `engine = "dense"`, `max_dense_bytes` bounds the dense algebra that the other discrete limits imply, and is checked before anything is allocated. The estimate covers the random-design matrix, the conditional Hessian and its factor, the slices of the design matrix that forming that Hessian materializes, the linear predictor and gradient, and a planning multiplier for copy-on-modify. It is not peak resident memory: it excludes the caller's data, the optimizer's own state, and the allocation the finite-difference stationarity pass repeats, so it underestimates what a fit really uses. Its purpose is to refuse the obviously impossible with a message naming the observation count, the random dimension, and the size of the random-design matrix, rather than to certify that a permitted model is comfortable. `gt_preflight()` reports the same estimate and its breakdown under `$resources`. Raising this limit makes neither the dense algebra practical nor the approximation accurate. For `engine = "sparse"`, `max_sparse_bytes` applies instead, with the sparse estimate just described.

Optional discrete information controls are:

- `check_hessian`: `FALSE`; `TRUE` requests observed information after numerical acceptance.
- `hessian_step`: `1e-3`, the finite-difference step multiplier in stored parameter coordinates, scaled by `max(1, abs(parameter))`. These coordinates include ordinal log gaps and are unaffected by optimizer scaling.
- `hessian_tolerance`: `0.05`, the maximum normalized disagreement between Hessians computed at the requested step and half that step.

On the finite-difference route the Hessian is formed from second differences of the objective, about $2p^2$ evaluations per step for p free parameters. With `gradient = "analytic"` it is formed instead from central differences of the analytic gradient, $2p$ gradients per step at cold conditional modes under the validation settings, and the difference matrix is symmetrized after its asymmetry, measured on the same normalized scale, is checked against `hessian_tolerance`. Both methods use the same steps and the same step-discrepancy, eigenvalue and inverse checks. A refused gradient makes information unavailable with its reason; the other method is not substituted. `fit$uncertainty` records `hessian_method`, `hessian_asymmetry` and the number of information gradients. The Hessian is for the full Laplace negative log likelihood, including nuisance intercepts and threshold coordinates together with covariance parameters. Source-covariance sampling uncertainty is obtained from the inverse of this full information matrix through the covariance-parameter Jacobian; nuisance parameters are not held fixed. A source covariance on or near its boundary (smallest eigenvalue below `1e-6`) has no ordinary Wald error. That source is held fixed at its fitted covariance and the information is formed over the remaining coordinates, as the Gaussian engine does when its joint curvature is indefinite. `fit$uncertainty` then records `restricted_to_interior`, `fixed_components` and `fixed_component_kinds` (`"zero"` or `"singular_nonzero"`), and the held source reports no standard error. Unusable conditional modes, parameter-bound contact, every estimated source on its boundary, unstable finite differences, or non-positive-definite information make Wald uncertainty unavailable. The reason is retained without changing an otherwise accepted point estimate. For a restricted calculation, the resulting coefficient interval treats the listed matrices as known. Its uncertainty excludes both their estimation and the selection of which sources to hold. A singular nonzero covariance is not replaced by a zero matrix. These are asymptotic intervals conditional on the first-order Laplace approximation; calibration after boundary selection needs a dedicated study. Saved information-only computations verify arithmetic at the retained estimates, not a fresh fit or coverage on another platform.

For p stored parameters, not counting any held fixed at a boundary, the two central-difference Hessians require $2(1 + 2p^2)$ marginal likelihood evaluations when all evaluations complete. These evaluations start their conditional modes cold. Information can therefore cost much more than point

estimation, particularly for joint categorical models. Full-panel fitting or a feasible sparse factorization does not establish that information or interval-pilot calculations are practical at that size.

Optimizer completion, numerical acceptance, and approximation adequacy are separate. Tighter numerical tolerances do not establish the adequacy of a first-order Laplace approximation.

Value

A `gt_control` object with separate Gaussian and discrete control lists and a resolved retention record. `print()` lists the settings that differ from the package defaults and the components a fit will keep, and returns the object invisibly.

See Also

`gt_diagnostics`, `gt_fit`

Examples

```
gt_control(gaussian = list(retry_seed = 123, check_hessian = FALSE),
  discrete = list(max_random_dimension = 100, alternative_starts = 1))
# Auto selects boundary-capable coordinates when the covariance model permits.
gt_control(discrete = list(covariance_parameterization = "auto"))
gt_control(discrete = list(optimizer = "nlminb", alternative_starts = 2))
# A lean fit for a large panel: estimates, diagnostics, reliability and
# decision studies are unchanged; the stored copies of the data are not kept.
gt_control(retain = list(data = FALSE, model = FALSE))
```

---

gt_design *Declare a Generalizability-Theory Random-Source Design*

Description

Declares an object of measurement, any number of instrumentation facets, selected object interactions, and explicit parent-scoped nesting. This constructor preserves the requested source terms; fitting validates the observed data and resolves any residual alias.

Usage

```
gt_design(object, facets, crossed = facets, nested = NULL,
  item_interactions = "complete", instrument_interactions = "complete",
  item_nested = character(), random = NULL, full_cell = TRUE,
  replicates = 1L, max_terms = 4096L, batch = NULL)
## S3 method for class 'gt_design'
print(x, ...)
```

Arguments

object	One column name for the object of measurement.
facets	Unique instrumentation column names, excluding the object.
crossed	A nonempty subset of facets allowed to form direct object interactions. Defaults to all facets.
nested	A named list giving the instrumentation parents of every non-crossed facet. Multiple parents define their joint group. Ancestors are expanded recursively; cycles are rejected.
item_interactions	"complete", "additive", or the maximum number of crossed instrumentation facets in an object interaction. The object itself is excluded from this order. Additive adds individual object-by-crossed-facet interactions.
instrument_interactions	"complete", "additive", or a maximum interaction order among crossed instrumentation facets. Declared nested groups remain present.
item_nested	Nested child names whose complete ancestor-expanded group should also interact with the object. This does not implicitly add other ancestor-by-object interactions.
random	Optional explicit one-sided formula or character vector of source terms. Formula syntax is limited to names, +, :, *, /, parentheses, and 0/1; a/b expands to a + a:b and (a + b)/c to a + b + a:b:c, as in R's formula algebra. Cannot be combined with explicitly supplied automatic selectors.
full_cell	Retain the object-by-all-facets source when the requested expansion contains it. FALSE removes exactly that one term and leaves every other requested source unchanged. Unlike the other selectors this may be combined with random.
replicates	Positive integer observations per complete object-by-facets cell. Observed replication is checked during fitting.
max_terms	Positive integer bound on source expansion, checked before expansion.
batch	Optional gt_batch declaration that items were annotated several to a call. NULL, the default, leaves items independent given the declared sources. Its by must name a declared facet, and its order and id columns other than the object and the facets.
x	A gt_design object for the print method.
...	Further print arguments; currently unused.

Details

The object main effect is always present. With all facets crossed, complete object and instrumentation interactions retain the full hierarchy. At one observation per cell, a Gaussian full-cell source is indistinguishable from a free residual and is combined with it during fitting. A requested observation-level source is rejected for discrete families, because it cannot be separated from the identified discrete observation model. It is never dropped silently: declare the reduced model with full_cell = FALSE, which removes only that source, or write the intended terms with random.

The same source grammar is accepted as the right-hand side of a two-sided model formula by gt_fit and gt_preflight, which then build this object themselves: `score ~ item * rater` is `gt_design("item", "rater")` for the outcome `score`.

Nesting defines parent-scoped groups shared across objects. It does not establish that a physical sampling design is balanced or identifiable. Arbitrary design declaration does not imply that every backend supports every resulting model.

Balanced coded panel. The default Gaussian route and ordinary analytic coefficients require one. It is the complete Cartesian product of the observed coded levels of the object and of every instrumentation facet, carrying exactly `replicates` rows in each of those cells, with no missing outcome. Nothing is imputed, aggregated or dropped to reach it. The explicit univariate lme4 option `panel = "observed"` can fit missing whole cells within its resource and covariance-identification limits; see gaussian_observed. It does not provide reliability for the irregular observed allocation. A separately requested `gt_dstudy(projection = "balanced")` projects its fitted components to an intended complete allocation under stated assumptions.

Coded is what makes a nested facet fit this description. A nested child is identified by its code within its parent, so child codes repeat across parents and the Cartesian product stays complete; the child's real identity is then the child-by-parent source, not the child column on its own. Globally unique child labels describe the same study but leave most child-by-parent cells structurally empty, and the default balanced backend rejects that panel instead of guessing which cells were never possible. The observed-row option still requires the declared source kernels to be identifiable and does not infer a physical sampling design. The example below shows both codings of one six-item design.

Value

A `gt_design` list containing the object, facets, requested terms and their members, nesting metadata, and pending data-validation information. `print()` shows the object, the facets, the requested sources, any nesting and batch declaration, and whether data have been checked against the declaration, and returns the design invisibly.

See Also

gt_fit, gt_family, gt_batch

Examples

```
complete <- gt_design("item", c("evaluator", "prompt", "temp", "seed"))
length(complete$terms_requested)
selected <- gt_design("item", c("evaluator", "prompt", "temp", "seed"),
  crossed = c("evaluator", "prompt"),
  nested = list(temp = c("evaluator", "prompt"), seed = "temp"))
selected$terms_requested
reduced <- gt_design("item", c("rater", "occasion"), random = ~ item + rater)
reduced$terms_requested
print(reduced)
# The same crossed design without the observation-level source, which a
# binary or ordinal fit at one observation per cell cannot identify. This is
# where a first discrete fit should start; see the worked script in
# system.file("examples", "discrete-first.R", package = "Gtheory4LLM").
discrete_ready <- gt_design("item", "rater", full_cell = FALSE)
```

```
discrete_ready$terms_requested

# Globally unique child labels versus within-parent codes, for six items
# nested in two questions and rated by four people.
coded_cells <- function(data, columns)
  prod(vapply(data[columns], function(x) length(unique(x)), integer(1)))
physical <- expand.grid(person = 1:4, item = 1:6)
physical$question <- 1 + (physical$item > 3)
# Unique labels 1-6 imply 48 coded cells but supply 24: item 4 never appears
# in question 1, so the coded panel is structurally incomplete.
c(rows = nrow(physical),
  cells = coded_cells(physical, c("person", "item", "question")))
physical$item_code <- physical$item - 3 * (physical$question - 1)
# The within-parent code describes the same six items as a complete panel.
c(rows = nrow(physical),
  cells = coded_cells(physical, c("person", "item_code", "question")))
nested_design <- gt_design("person", c("item_code", "question"),
  crossed = "question", nested = list(item_code = "question"))
nested_design$terms_requested   # item_code:question identifies the six items
nested_design$nested_groups
# The same nesting written as a source formula: question / item_code is
# question + question:item_code.
gt_design("person", c("item_code", "question"),
  random = ~ person + question / item_code + person:question)$terms_requested
```

`gt_diagnostics` *Inspect Numerical Acceptance and Backend Diagnostics*

Description

Reports optimizer completion separately from numerical acceptance and likelihood-approximation adequacy.

Usage

```
gt_diagnostics(fit)
## S3 method for class 'gt_diagnostics'
print(x, ...)
## S3 method for class 'gt_diagnostic_stages'
print(x, ...)
## S3 method for class 'gt_diagnostic_stage'
print(x, ...)
```

Arguments

`fit`	A `gt_fit` object.
`x`	A `gt_diagnostics`, `gt_diagnostic_stages` or `gt_diagnostic_stage` object.
`...`	Reserved for additional print arguments.

Details

Gaussian diagnostics include independent likelihood reconstruction, covariance-scale stationarity, boundary components, and optional Hessian diagnostics. Discrete diagnostics include covariance- and parameter-scale stationarity, bound locations, and tighter-tolerance alternative-start stability. A discrete fit also retains `evaluations`: how many objective evaluations the optimizer made, how many it could not use, how many valid conditional solves ended at the inner iteration budget, and, for the first eight invalid evaluations and for later solve- or factor-validity events until eight such events are held, the parameters they were asked about and the measurements each evaluation itself reported. These counts and records cover the optimizer's logged evaluations; a final check or a stationarity probe that returns a refusal keeps its own record with that check. Matrices are never retained in the fit. Setting the environment variable `GTHEORY_DISCRETE_SPECIMEN_DIR` to a directory writes one specimen file per retained failure and per refused start, holding the failed operation's matrix, right-hand side, step and factor where the failure produced them, its parameters and measurements, and the numerical environment, but no observations. The conditional-mode stage reports the three counts, and the count of validity events, among its measurements. Passing these checks does not prove structural identification, a global optimum, or adequate approximation in sparse or strongly dependent data.

For discrete fits, the engine detail record `diagnostics$parameter_domain` retains the requested, default and effective optimizer-coordinate domains described in `gt_control`. These declarations do not alter the separate artificial-contact rejection.

`selected_attempt` identifies the returned candidate using the engine's recorded trial identifier. It is `NA` when no trial is recorded as having produced the returned fit. `acceptance_failures` gives reasons for rejecting the returned fit; `attempt_failures` is a two-column data frame of recorded attempt identifiers and error or rejection reasons, including earlier attempts. An empty attempt-failure table means no recorded trial reported an error or a rejection reason; it is not a statement about trials that were never run.

`standard_errors_available` is `TRUE` only when the selected engine obtained a usable parameter covariance under its curvature policy. The OpenMx Gaussian engine forms the parameter covariance matrix as `2 * H^-1` for the `-2 log L` fit function and maps it to source covariance entries by the delta method; `fit$uncertainty` retains that matrix, the entry Jacobian, and the entry covariance matrix used by `gt_reliability()`. The lme4 engine returns source covariance estimates but does not compute their sampling covariance or source-variance standard errors. Optional discrete full observed information uses `H^-1` on the negative-log-likelihood scale, retaining nuisance intercepts and thresholds when extracting source-covariance uncertainty. It requires numerical acceptance, stable finite differences, and positive-definite, sufficiently conditioned information; a source on or near its boundary is held fixed at its fitted covariance and the information is formed over the remaining parameters. Otherwise the recorded reason explains why it is unavailable. Its asymptotic intervals remain conditional on first-order Laplace and do not establish finite-sample coverage. A component at a variance boundary is flagged in `component_standard_errors`: Wald theory does not apply there, and the reported number is not a usable standard error.

For lme4 fits, the retained theta values are relative standard deviations (source standard deviation divided by residual standard deviation). The theta vectors, derivative vectors, and Hessian rows and columns preserve lme4's order. Their position `i` refers to row `i` of `parameter_map`; the integer positions in `stationarity$boundary` select its boundary rows. Within `stationarity`, the `boundary_one_sided_derivatives` entries follow those selected rows in the same order. The corresponding source rows are obtained with:

```
parameter_map[stationarity$boundary, ]
```

No parameter or boundary index is reordered by the map.

`boundary_sources` describes natural zero or nearly singular covariance components and does not by itself imply rejection. `parameter_bounds` lists artificial discrete-parameter bound contacts, which do prevent numerical acceptance. These shared fields summarize existing engine decisions without repeating or changing numerical checks. `summary(fit)` prints a concise version while retaining full diagnostics for programmatic inspection.

Printing a `gt_diagnostics` object reports the same decisions in the same words. It reads recorded fields only: no check is re-run, no rejection is re-evaluated, and every field remains available in the returned list.

Value

A `gt_diagnostics` list with estimator, engine, optimizer and numerical acceptance status, approximation status, engine-specific diagnostics, a staged summary, resolved aliases, data-validation notes, and `batch`: whether the design declared batches and whether the fit modelled them, their size, whether membership was inferred or recorded, the reason a declared batch was not modelled, and the declared options not yet estimated.

The `stages` element summarises the numerical checks a fit passed through, so that a rejection names the stage responsible without the caller reading private optimizer records. Each stage reports `status`, a `reason` and supporting `measurements`. The status is one of `"passed"`, `"failed"`, `"not_assessed"` for a check that did not run or does not apply to this engine, or `"inconclusive"` for one that ran and neither clearly passed nor clearly failed. Absent evidence is reported as `"not_assessed"` and never as a pass. The summary is derived from evidence the fit already retained: it never refits, and it never changes an acceptance decision. In particular, optimizer completion reports what the optimizer reported; a retained attempt that never moved from its starting values is exposed through `max_abs_parameter_change` rather than by redefining completion. Shared diagnostic fields include:

- `engine`, `optimizer`, `approximation`: the fitting engine, its optimizer, and the likelihood approximation. The engine is the exact balanced Gaussian engine, the lme4 Gaussian engine, or the selected dense or sparse discrete Laplace engine, each under its own label; the approximation is none for a Gaussian engine and first-order Laplace for a discrete one. They are recorded separately because choosing another optimizer is not choosing another engine, and neither changes the approximation.
- `optimization_trials`: the recorded trial count.
- `selected_attempt`: the returned candidate's identifier.
- `acceptance_failures`: reasons for rejecting the returned fit.
- `stationarity`: the record of the independent stationarity check that decided acceptance, with its verdict, tolerance, and the scaled gradient it measured.
- `likelihood_agreement`: for the **lme4** engine, the independent Gaussian deviance at the estimates, the engine's criterion and their difference. The reference value is stored under the name for the fitted panel:
  - `exact_balanced_deviance`: complete-panel route.
  - `exact_observed_row_deviance`: observed-row route.

The agreement record is NULL for engines that make no such comparison.

- gradient: the outer gradient record of a discrete fit. Its route names the gradient the optimizer attempts received, finite differences by default or the analytic gradient on request, and the analytic route also records the number of gradients, refusals with the first eight retained, zero gradients returned at unusable objective points, gradients whose objective record had to be evaluated first, seconds by stage, the structural envelope and selected-inverse sizes, the platform's long-double width, and, as a diagnostic that acceptance never reads, the analytic gradient at the selected point beside the stationarity check's finite-difference fixed-effect score. NULL for other engines.
- attempt_failures: recorded trial errors or rejections.
- boundary_sources: zero or nearly singular covariance sources.
- parameter_bounds: artificial parameter-bound contacts.
- parameter_map: for the lme4 engine, a data frame with integer index, scientific source, and engine_parameter identity in lme4's parameter order; NULL for engines without this map. It remains available when the fitted model and data are not retained.
- initialization: for the exact Gaussian engine, the square-root recovery record described in gt_control: the requested policy and, once invoked, the algorithm; a status taking one of the five values listed in gt_control (disabled, not applicable, not invoked, handoff eligible, refused); the reason where one applies; the trigger and handoff attempts and the ordinary budget; the initializer's optimizer convergence code, message and evaluation counts; its objective, the independent deviance and their difference; and elapsed seconds. NULL for other engines.
- standard_errors_available: whether a usable parameter covariance matrix was obtained.
- standard_errors_unavailable_reason: why it was not, when it was not.
- component_standard_errors: per-source variance standard errors with a boundary flag.

See Also

gt_control, gt_fit

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
gt_diagnostics(fit)$numerically_accepted
print(gt_diagnostics(fit))
```

gt_dstudy *Evaluate Balanced Measurement Allocations*

Description

Computes supported G and Phi coefficients over a grid of instrumentation counts, validating the fitted context once.

Usage

```
gt_dstudy(fit, grid, scale = NULL, score = NULL, fixed = character(),
  level = 0.95, projection = NULL, bootstrap = NULL, interval_type = NULL)
## S3 method for class 'gt_dstudy'
print(x, ..., digits = 4L, max_rows = 12L)
## S3 method for class 'gt_dstudy'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
gt_dstudy_target(x, target, coefficient = "Erho2", outcome = NULL,
  kind = "outcome")
## S3 method for class 'gt_dstudy'
plot(x, coefficient = "Erho2", interval = TRUE, ...,
  target = NULL, outcome = NULL, kind = NULL)
```

Arguments

fit	A numerically accepted gt_fit with a complete balanced coded panel.
grid	A nonempty data frame of positive integer facet counts. Columns name declared facets; omitted facets retain their fitted counts.
scale	"observed" for Gaussian outcomes or explicitly "latent" for binary, ordinal and nominal outcomes; see gt_reliability for the nominal coefficients.
score	Optional composite weights created by gt_score().
fixed	Instrumentation facets treated as fixed, as in gt_reliability(). A fixed facet may not appear in grid: its universe is exactly its observed levels.
level	Confidence level for the reported coefficient intervals.
projection	Normally NULL. Set "balanced" for an observed-panel univariate Gaussian fit to project an intended complete balanced allocation. Supply every facet in grid and keep all facets random. This does not describe the reliability of the irregular observed averages. The projection_target export identifies the future allocation. The projection_assumptions export states its assumptions.
bootstrap	Optional gt_bootstrap result for this fit. Each allocation's standard errors and intervals then come from the replicates' projections to that allocation, as in gt_reliability. It cannot be combined with projection.
interval_type	"wald" without a bootstrap; "percentile" (default) or "bca" with one.
x	A gt_dstudy object.

coefficient	Plot or screen "Erho2" (default) or "Phi".
interval	Draw the interval bounds as vertical segments when the fit supplies them.
target	One finite coefficient threshold from 0 to 1, chosen and justified by the user. Required for screening. Optional in plotting, where NULL omits the reference line.
outcome	One exact reported outcome name. Screening requires it when more than one outcome exists within the selected kind. Plotting defaults to all outcomes.
kind	"outcome" for measured outcomes or "composite" for the weighted composite. Screening defaults to "outcome". Plotting defaults to both kinds; specify the kind when an explicitly selected outcome label also names a composite.
row.names	Optional row names for the exported data frame.
optional	Accepted for generic compatibility. Exported column names are always preserved without syntactic name repair.
digits	Significant digits used when printing.
max_rows	Maximum number of coefficient rows to print.
...	Additional arguments passed to graphics::plot(). A named argument such as col, pch, ylim, xlab or ylab replaces the corresponding default rather than duplicating it. Reserved in the print and data-frame methods.

Details

Tabular export. as.data.frame() returns an ordinary data frame joining each coefficient row to its allocation by the recorded design_id, even for multivariate results or reordered coefficient rows. It preserves coefficient-row order. kind remains "outcome" or "composite"; a measured outcome named "composite" is distinct from a weighted composite. Export fields follow the reliability data-frame method documented in gt_reliability. All columns are atomic vectors suitable for write.csv(); additional R attributes must be retained separately when exporting to CSV.

Screening supplied candidates. gt_dstudy_target() returns an ordinary data frame with every supplied allocation for the selected outcome and kind, preserving coefficient-row order. The added fields are:

- coefficient and target: the chosen criterion.
- estimate: the selected coefficient value.
- meets_target: the point-estimate comparison.
- fewest_measurements: the qualifying minimum-count flag.

meets_target compares the point estimate with the target using >=. It does not screen a confidence limit. An unavailable estimate gives NA, not FALSE. Among candidates known to meet the target, fewest_measurements marks all ties at the smallest measurement count, including duplicated supplied allocations. Unknown coefficients retain NA in this flag. If no known candidate meets the target, no row is marked TRUE. The target_context attribute records the chosen coefficient, target, outcome, kind, scale, point-estimate comparison, restriction to supplied candidates, and batch status. Other export context is retained, including the batch_status column: a screen of a design that declared batches is a screen of coefficients that do not model them.

Counts at or above 2^53, or overflowing to infinity, cannot be ranked as exact integer totals. Their measurement_count_exact flag is FALSE. If no qualifying candidate has a smaller exactly represented total, fewest_measurements remains NA for these qualifying candidates rather than reporting a potentially false tie. If a qualifying exact small total exists, these larger totals cannot be the minimum.

This helper does not generate allocations, refit models, use monetary costs, or find a global optimum. The target is a criterion applied to conditional point projections, not a guarantee of reliability in a new panel. Examine available intervals, extrapolation flags, and the model's generalization assumptions alongside the screen. Default latent discrete delta-method intervals require the optional accepted, stable full-information calculation. An explicit gt_bootstrap supplies object-bootstrap intervals under its own availability checks and conditional interpretation.

Plotting. By default allocations are unconnected points against measurements per object; different allocations can have the same count. Colours identify outcome/composite series, and open symbols distinguish allocations exceeding fitted facet counts from filled symbols within them. An optional target appears as a dashed horizontal line and is included in automatic vertical limits. Explicit graphical settings remain authoritative, including pch and ylim. For a design that declared batches with gt_batch, the plot states above the plotting region that they are not modelled, whatever subtitle is supplied. Series filters do not change the returned object: plotting returns the full original study invisibly.

Balanced coded panel. This is the complete Cartesian product of the observed coded levels of the object and of every instrumentation facet, carrying exactly the declared number of rows in each of those cells, with no missing outcome. For a nested facet, *coded* means the child's within-parent code rather than a globally unique label, so the child's identity is the child-by-parent source; gt_design works both codings through. A panel that is not balanced in this sense has no analytic coefficient here, and none is approximated.

These are conditional design projections using fitted source covariances. They do not refit the model or establish that larger allocations preserve the same source distributions. Default delta-method intervals propagate estimation uncertainty in the fitted source covariances under the declared random-effects sampling model. They are not prediction intervals for the realized scores of a newly drawn panel, and they do not quantify failure of exchangeability when extending to new facet levels. Boundary and conditional-inference caveats are retained from the fitted model. Default latent discrete delta-method intervals inherit the Laplace approximation and require accepted, stable observed information. Object-bootstrap intervals instead describe object sampling conditional on the observed instrumentation levels and numerical acceptance; they do not require observed information. Restrictions and coefficient definitions are those of gt_reliability().

A nested child's count may be projected like any other random facet. Its own source and every source containing it are then divided by the joint count of child and parent, while sources built from the parent alone keep the parent count: adding children therefore reduces less error than adding an equal number of levels of a fully crossed facet. A fixed facet may not appear in grid, but its random children still may.

Value

A gt_dstudy list with completed allocation rows, coefficient results including standard errors and interval bounds, measurements per object, scale, fixed facets, the uncertainty record, extrapolation flags, and fitting diagnostics. The print method shows allocations and coefficients without expanding fitting diagnostics. Both methods return the object invisibly.

See Also

gt_reliability, gt_score

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
study <- gt_dstudy(fit, data.frame(rater = c(2, 3, 6)))
as.data.frame(study)
gt_dstudy_target(study, target = 0.80, coefficient = "Phi")
plot(study, coefficient = "Phi", target = 0.80)
```

---

gt_example *Load Modeling Tables from Three LLM-Annotated Datasets*

Description

Lists eight documented outcome sets or loads their design columns and modeled outcomes. Installed resources omit raw text and API metadata.

Usage

```
gt_example(name = NULL, coding = c("native", "numeric", "manuscript"),
  directory = NULL)
## S3 method for class 'gt_example'
print(x, ...)
```

Arguments

name	Outcome-set name, or NULL to list available sets.
coding	Use "native" to preserve response types. Use "numeric" for the category codes as numbers with Gaussian working-score families, available for seven sets. "manuscript" is the deprecated name of "numeric": it returns the same tables with a warning.
directory	Optional directory containing example RDS resources or source CSV files, supplied explicitly as one nonempty path. A file named <name>.rds takes precedence over CSV loading. An installed package always loads bundled RDS resources when this argument is omitted, regardless of workspace variables.
x	A gt_example object for the print method.
...	Further print arguments; currently unused.

Details

Load these installed resources with `gt_example()`, not `data()`. The loader keeps the chosen outcome coding, its family specification, and source provenance together with the measurement table.

Every set retains 21,600 measurement rows: 100 text objects by 4 evaluators, 3 prompts, 6 temperatures, and 3 seeds. Full tables exceed default discrete observation and random-dimension limits. The sparse engine changes the matrix calculations without relaxing those guards or qualifying full-panel fits; use a predeclared small panel for bounded discrete examples.

Available outcome sets are:

- `hate_speech`: ordinal severity.
- `mental_health_7L`: original Gaussian working score.
- `mental_health_3L`: original Gaussian working score.
- `mental_health_6flag`: six binary flags.
- `mental_health_3group`: three binary flag groups.
- `drug_review`: ordinal holistic sentiment.
- `drug_review_4aspect`: four ordinal aspects.
- `mental_health_nominal`: seven unordered categories.

The nominal set has no numeric coding. Detailed variable definitions, category mappings, preprocessing notes, and the corresponding annotation-paper and original-corpus references are provided in the Hate-Speech, Mental-Health, and Drug-Review data topics. The dataset overview documents the shared panel design, provenance, and installed bibliography.

Standalone functions can use a local `.gt_example_data_dir` binding to configure the CSV directory. This binding must be in the same environment into which the functions were sourced. The source loader configures the checkout's public `inst/extdata` directory. Parent environments and callers are not searched. This standalone convention has no effect on the installed package; use `directory` for an intentional override. An invalid explicit directory produces an error rather than silently falling back to bundled data.

The mental-health 7L/3L scores are study-defined working scores, not established clinical severity scales. Labels are model-generated and retain the source study's preprocessing; no new human adjudication is implied.

The source CSV files are:

- '`hate_labeling_final.csv`'
- '`mh_labeling_final.csv`'
- '`drug_labeling_final.csv`'

Bundled data retain design identifiers and modeled outcomes only. Source checksums, the public OSF deposit link, and extraction details accompany installed resources. The annotation data retain the CC BY 4.0 license stated in the public data codebook; see the installed '`DATA_LICENSE.md`'.

Value

A catalog data frame when name is NULL. Otherwise a `gt_example` list with data, outcome names, observation families, object and facet names, example name, coding, source, and interpretation notes. Installed resources also provide portable source-provenance metadata. `print()` describes the set without printing any rows and returns the object invisibly.

See Also

Dataset overview, Hate-Speech data, Mental-Health data, Drug-Review data, `gt_family`, `gt_fit`

Examples

```
gt_example()
x <- gt_example("hate_speech")
nrow(x$data)
is.ordered(x$data$score)
nominal <- gt_example("mental_health_nominal")
levels(nominal$data$label)
```

`gt_family` *Specify a Gaussian, Binary, Ordinal, or Unordered Categorical Outcome*

Description

Keeps observation families distinct and defines their links and category interpretation.

Usage

```
gt_family(family = "gaussian", link = NULL, levels = NULL,
  reference = NULL)
## S3 method for class 'gt_family'
print(x, ...)
```

Arguments

`family`	One of `"gaussian"`, `"binary"`, `"ordinal"`, or `"categorical"`.
`link`	Gaussian uses identity; binary and ordinal use probit (default) or logit; unordered categorical uses softmax.
`levels`	Distinct category labels. Binary levels are negative then positive. Ordinal levels give the substantive order. Categorical levels identify categories without assigning an order. Gaussian outcomes have no category levels.
`reference`	One unordered categorical reference category. Defaults during fitting to the first declared category.
`x`	A `gt_family` object for the print method.
`...`	Further print arguments; currently unused.

Details

The package uses `gt_family()` rather than base R family objects to record category ordering and reference categories as part of the observation model. A named list assigns these specifications to joint outcomes; joint Gaussian–discrete fitting is not implemented.

Numeric binary responses must be 0/1. Ordinal and categorical families require at least three categories; use the separate binary family for two categories. Ordinal order must come from declared levels or an ordered factor. Every declared category must occur in the fitted data. Numeric Gaussian outcomes are never constructed by silently converting factors.

Categorical random effects use reference-category contrasts. An unstructured covariance can be transformed under a reference change; a diagonal contrast covariance imposes a model restriction that depends on the reference category.

Value

A `gt_family` object describing the observation distribution. `print()` shows the family, link, declared levels in their order and reference category, and returns the object invisibly.

See Also

`gt_fit`

Examples

```
gt_family()
gt_family("binary", link = "logit")
gt_family("ordinal", levels = c("low", "middle", "high"))
gt_family("categorical", levels = c("red", "blue", "green"), reference = "blue")
```

---

gt_fit *Fit Univariate or Joint Multivariate Generalizability Models*

Description

Fits selected random-source covariance models using exact Gaussian likelihoods or dense and sparse first-order Laplace likelihoods for discrete outcomes.

Usage

```
gt_fit(data, outcomes, design = NULL, family = gt_family("gaussian"),
  estimator = NULL, covariance = "unstructured", residual = NULL,
  control = gt_control(), formula = NULL, measured = NULL)
## S3 method for class 'gt_fit'
print(x, ...)
## S3 method for class 'gt_fit'
summary(object, ...)
```

Arguments

data	Data frame containing the object, all declared instrumentation columns, and outcomes. Missing values are not silently dropped. With a model formula the data frame may also be given second, after the formula.
outcomes	One or more unique outcome column names, or a two-sided model formula. The formula names the outcomes on its left, one column or cbind() of several, and the random sources on its right in the grammar of gt_design()'s random. The object of measurement is the variable named by measured, or else the first variable on the right; the other variables are the instrumentation facets. score ~ item * rater declares the sources item + rater + item:rater; label ~ item + rater is the reduced model a one-observation-per-cell discrete fit needs.
design	A declaration created by gt_design(). Omit it with a formula, which supplies the design; a family given in this position after a formula is read as family, the order glm() users expect. Replicates, batches and the automatic nesting selectors have no formula syntax and need gt_design(). A facet absent from every random-source term also needs an explicit design declaration.
formula	The model formula under its own name, for calls such as gt_fit(formula = score ~ item * rater, data = d); the same as giving it as outcomes.
measured	With a formula, the name of the right-hand variable that is the object of measurement. NULL takes the first variable on the right, and the fit's design records which rule applied. Re-ordering a formula without measured changes the object of measurement, and with it every reliability coefficient, while the variance components stay the same; name it whenever the order is not deliberate. The argument is not called object because that is the first argument of update() and the other model generics.
family	One gt_family object for all outcomes, or a named list matching outcomes exactly. Multiple discrete families can be modeled jointly. Joint Gaussian-discrete likelihoods are not implemented.
estimator	Gaussian: "REML" (default) or "ML". Discrete: "ML_Laplace".
covariance	"unstructured" or "diagonal"; Gaussian also accepts named source overrides, with unspecified sources using diagonal covariance. Discrete source covariance uses the common requested structure. A discrete fit with a nominal outcome also accepts "exchangeable": each source gives every category its own random utility with one variance, $\sigma^2(I + 11')$ on the K - 1 contrasts, unchanged by a change of reference category, with one variance per source and outcome; binary and ordinal outcomes in the same fit keep one variance each, and outcomes are independent within a source.
residual	The Gaussian residual structure. Choose "unstructured", "diagonal", or "pooled"; pooled gives equal outcome variances. Omit for discrete outcomes.
control	A gt_control object.
x, object	A fitted gt_fit object for the print or summary method.
...	Further method arguments; currently unused.

Details

Formula interface. `outcomes` may be a two-sided model formula, with the data frame given before or after it. `gt_fit(score ~ item * rater, data)` declares the outcome `score`, the object of measurement `item`, the facet `rater`, and the sources `item`, `rater` and `item:rater`. A joint fit of two outcomes is written `cbind(efficacy, safety) ~ item + rater`. Nothing in the formula is evaluated: it names columns and sources, and `fit$design$formula` records it. The resolved design is the object `gt_design(object, facets, random = ~ ...)` builds, so a formula fit and a design fit of the same sources are the same fit. Every fit, however declared, answers `formula()` and `update()`; see `gt_fit_methods`. The recorded `fit$call` names each argument, with the formula under `formula` and the data under `data` whichever order they were given in.

Engines. Gaussian outcomes use the exact balanced likelihood through OpenMx by default. Set `engine = "lme4"` in the Gaussian controls to fit the same sources as random intercepts through lme4. The criterion is cross-checked against the exact balanced deviance.

Adding `panel = "observed"` to those controls opts into bounded univariate fitting on incomplete observed rows. This route is checked against an independent dense covariance likelihood; see gaussian_observed.

The discrete controls default to `engine = "dense"`; use `engine = "sparse"` for sparse algebra. Both calculate the same first-order Laplace objective for binary, ordinal and nominal outcomes, separately or jointly, and use the same acceptance policy. Sparse algebra does not refine the likelihood approximation or certify large-panel performance. Sparse L-BFGS-B fits can request `gradient = "analytic"`, which replaces the optimizer's finite-difference gradient with the analytic gradient of the same objective; the acceptance policy is unchanged, and `gt_control` lists the supported covariance coordinates and the refusals.

`gt_laplace_check` attempts a fixed-parameter numerical comparison with a refined marginal likelihood and reports diagnostic precision and usability; finite-sample accuracy is not guaranteed. The diagnostics record the engine, its optimizer and the approximation separately.

The default Gaussian route requires a complete coded Cartesian panel with one observation per cell and at most 4096 factorial strata. A design that declares equal fixed batches with `gt_batch` adds one source, `Call`, shared by the items of a batch under one condition, and one factorial axis, because an item is then a batch-by-slot cell; see `gt_batch` for when a declaration is modelled. Implicit normalized Helmert contrasts avoid quadratic allocation along a large object axis. Gaussian ML profiles one fixed intercept per outcome; balanced REML includes the unscaled-intercept constant D * log(N). The observed-row option instead evaluates $\log |X'V^{-1}X|$ on the retained rows, with an unscaled intercept, and refuses batch declarations. Gaussian ML AIC counts covariance parameters and profiled means. Its generic BIC uses N response vectors; `ml_BIC_scalar_scores` explicitly provides the alternative N * D convention. The exact balanced engine preserves the archive's variance-only parameter convention in `legacy_AIC` and `legacy_BIC`; these fields are unavailable on the lme4 route. For REML the fit's `AIC` and `BIC` elements are `NA`; `AIC()` and `BIC()` use the restricted likelihood with covariance parameters only, `BIC()` over N response vectors. The fit records the `AIC()` value, the `BIC()` value, and the alternative N * D scalar-score BIC under explicit names, in that order:

- `reml_AIC_variance_parameters`
- `reml_BIC_response_vectors_variance_parameters`
- `reml_BIC_scalar_scores_variance_parameters`

Discrete models use binary links, ordinal thresholds, or unordered reference-category softmax contrasts. They do not estimate a free Gaussian observation residual. Probit latent residual variance is identified as 1; logit as pi^2 / 3. Unsupported observation-level sources, aliased free source kernels, missing categories, and resource-limit violations are rejected.

Incomplete Gaussian fitting is univariate only. Discrete designs may omit whole cells, including for joint outcomes, provided retained cells have the declared equal replication and every retained row contains all requested outcomes. Partially observed outcome vectors are rejected. This does not model the missingness mechanism or provide analytic reliability for an irregular panel.

Both discrete engines can append a `Call` source for an eligible fixed batch layout. Its grouping is the batch under the complete combination of instrumentation facets. The call source is included in random-dimension, covariance-parameter, memory and kernel-identification checks. This does not implement variable batch sizes, positional dependence or automatic reliability for a shared-call layout.

Returned binary probabilities evaluate each tail directly, preserving small representable probabilities and the declared category order. These are conditional probabilities at the random-effect mode, not population-marginal probabilities.

The OpenMx Gaussian engine also computes asymptotic Wald standard errors for source variance components. These use a numerical Hessian of the restricted or profile likelihood. These are large-sample quantities conditional on the declared model; they do not establish coverage in small designs and do not apply to a component resting on a variance boundary. `check_hessian = FALSE` disables them along with the other derivative diagnostics on that engine. The lme4 engine returns estimated source covariance matrices but does not compute their sampling covariance or source-variance standard errors.

Discrete `check_hessian = TRUE` requests observed information of the Laplace negative log likelihood after acceptance. Nuisance intercepts and thresholds are included when covariance-parameter uncertainty is extracted. If an estimated source covariance lies on its boundary, the entire source matrix is held at its fitted value, which may be zero or singular and nonzero; information is then computed over the remaining coordinates. The resulting latent coefficient intervals are conditional on those held matrices, whose own standard errors remain unavailable. The uncertainty record names the held sources and their kinds. This does not account for uncertainty in selecting the boundary sources or estimating their held values.

Step-size instability, unusable conditional modes, parameter-bound contact, all estimated sources at their boundaries, or non-positive-definite information still withhold uncertainty with a reason. Where available, the delta-method errors and intervals are asymptotic and depend on the first-order Laplace likelihood; coverage after boundary selection has not been established. The analytic route differences gradients rather than objectives to form the information matrix. See `gt_control` for its coordinates and checks.

Numerical acceptance includes optimizer completion and independent stationarity, bound, and tighter-tolerance restart checks. First-order stationarity does not establish a global optimum. A numerically accepted Laplace fit still requires an assessment of approximation adequacy for the intended analysis. Reliability and decision studies reject fits that fail numerical acceptance.

Reproduction and empirical scope. Repeating the same request requires the recorded data and coding, source archive, controls, seeds and numerical environment; see gt_control. Across platforms, compare unrounded estimates under declared tolerances and require agreement in numerical-acceptance and uncertainty-availability decisions. Identical printed coefficients alone do not establish reproducibility, and bitwise-identical optimization is not promised.

The installed 'examples/native_1_0_workflow.R' defines a public workflow for two specified native examples:

- Two ordinal outcomes, efficacy and safety, from drug_review_4aspect.
- Three binary (yes/no) groups in mental_health_3group.

Both are loaded with gt_example() and coding = "native". Each full request retains 21,600 rows and ten declared sources. The project records report numerical acceptance on macOS and Linux for those requests, with sparse analytic-gradient fitting and projected boundary escape. Their original ordinary information was unavailable. Later macOS information-only computations at those saved estimates produced conditional intervals while holding named singular nonzero source matrices fixed. The original point references come from development source 2fc771b. On 28 October 2026 the candidate built from 2b31e84 was refit on both panels on macOS and Linux: all four fits were accepted and met both the point and conditional-information comparison criteria. The repository's release record identifies those exact archive bytes. That evidence predates the subsequent bootstrap data-association repair; it is not a full-data refit of a later archive and does not assess interval coverage. These records do not establish higher-order likelihood accuracy or results for the four-aspect drug-review and six-flag mental-health models. Sourcing the script does not fit the native panels; its helper requires an explicit mode = "full" call. Its small synthetic mode = "smoke" checks the reporting path only.

Value

A gt_fit object whose class vector is "gt_gaussian_fit" or "gt_discrete_fit", according to the engine, followed by "gt_fit". It contains source covariance matrices, estimates, likelihood and parameter counts, the resolved design and observation families, modeled data, and numerical diagnostics. Gaussian objects retain prepared sufficient statistics and the fitted engine model (OpenMx or lme4), subject to the retention controls. The OpenMx path also records source-covariance uncertainty and a component_standard_errors table when its curvature checks permit them. The lme4 path returns estimated source covariance matrices but does not compute the sampling covariance of their estimates; source-variance standard errors and delta-method coefficient intervals are unavailable. For a complete panel without declared batches, gt_bootstrap supplies object-bootstrap coefficient intervals. Discrete objects include thresholds or category contrasts and conditional probabilities evaluated at the random-effect mode. print() returns the fit invisibly. summary() returns a summary.gt_fit list with estimates, a source-variance table carrying standard errors and a boundary flag where they are available, full covariance components, for a Gaussian fit the observed-score reliability at the fitted facet counts as a gt_reliability object in reliability where the fit allows one and otherwise the reason in reliability_note, and diagnostics. Its print method displays a concise report of numerical acceptance, selected attempts, boundaries, and estimates.

See Also

gt_design, gt_control, gt_diagnostics. The package installs a worked first discrete fit as discrete-first.R, and the scoped native workflow as native_1_0_workflow.R.

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
```

```
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
print(fit)
deviance(fit)
# The same model as a formula: outcomes on the left, random sources on the
# right with the object of measurement first.
formula_fit <- gt_fit(score ~ item + rater, d, measured = "item",
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
all.equal(gt_components(formula_fit), gt_components(fit))
summary(formula_fit)
# The same model through lme4, cross-checked against the exact balanced
# deviance at its estimates; no standard errors on this path.
if (requireNamespace("lme4", quietly = TRUE)) {
  lme4_fit <- gt_fit(d, "score", design,
    control = gt_control(gaussian = list(engine = "lme4")))
  gt_diagnostics(lme4_fit)$engine
  all.equal(gt_components(lme4_fit), gt_components(fit), tolerance = 1e-4)
}
# Re-ordering the right-hand side would make rater the object of measurement;
# measured keeps the target explicit whatever the order.
gt_fit(score ~ rater + item, d, measured = "item",
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))$design
# A discrete outcome at one observation per cell: declare the design without
# the object-by-all-facets source, which no discrete fit there can identify.
b <- expand.grid(rater = 1:4, item = 1:10)
b$label <- as.integer((b$item + 2 * b$rater) %% 5 > 1)
binary_fit <- gt_fit(b, "label", gt_design("item", "rater", full_cell = FALSE),
  gt_family("binary"))
gt_diagnostics(binary_fit)$numerically_accepted
# As a formula, with the family following it as in glm().
binary_formula <- gt_fit(label ~ item + rater, b, gt_family("binary"),
  measured = "item")
all.equal(gt_components(binary_formula), gt_components(binary_fit))
# The installed worked example, start to finish:
script <- system.file("examples", "discrete-first.R",
                      package = "Gtheory4LLM")
file.exists(script)
```

`gt_fit_methods` *Standard Extractors for Fitted Generalizability Models*

Description

Log likelihood, deviance, observation count, model formula, refitting with `update()`, fixed location estimates, and the sampling covariance of the estimated source covariances, following the usual R conventions where they apply and refusing where they do not.

Usage

```
## S3 method for class 'gt_fit'
```

```
logLik(object, ...)
## S3 method for class 'gt_fit'
deviance(object, ...)
## S3 method for class 'gt_fit'
nobs(object, ...)
## S3 method for class 'gt_fit'
formula(x, ...)
## S3 method for class 'gt_fit'
update(object, formula., ..., evaluate = TRUE)
## S3 method for class 'gt_fit'
coef(object, ...)
## S3 method for class 'gt_fit'
vcov(object, ...)
gt_component_vcov(fit, type = c("components", "parameters"), ...)
```

Arguments

object, x	A fitted gt_fit object.
fit	A fitted gt_fit object.
formula.	Changes to the model formula in the notation of update.formula, such as . ~ . + item:rater; omit it to change other arguments only.
evaluate	Whether to refit. FALSE returns the updated call instead.
type	Which covariance matrix to return. "components" gives the delta-method covariance of the unique source covariance entries. Each is labelled by its source and matrix position. "parameters" gives the free optimizer-parameter covariance instead.
...	For update(), arguments of gt_fit to add or replace in the recorded call, for example estimator = "ML" or measured = "item". Otherwise unused.

Details

Parameter counts. Gaussian ML profiles one intercept per outcome. Those intercepts are estimated, so df counts them alongside the covariance parameters, and AIC() and BIC() then reproduce the fit's ml_AIC and ml_BIC_response_vectors. Gaussian REML uses a restricted likelihood whose df counts the covariance parameters only; the returned object sets REML = TRUE so nothing downstream mistakes it for an ML value, and AIC() reproduces reml_AIC_variance_parameters. BIC() on a REML fit uses N response vectors with that same parameter count, which the fit records as reml_BIC_response_vectors_variance_parameters; the N * D scalar-score convention remains reml_BIC_scalar_scores_variance_parameters. A restricted-likelihood criterion compares models sharing the same fixed-effects structure. Every fit here has exactly one intercept per outcome, so covariance models remain comparable, but a REML value must never be compared with an ML value.

Observations. nobs() counts measurement rows, which are the response vectors of the likelihood. A joint fit of D outcomes on N rows has N response vectors and not N * D independent observations, because the outcomes within a row share the same random sources. BIC() therefore uses N. The alternative N * D scalar-score convention and the archive's variance-only conventions remain in the fit object under explicit names; see gt_fit.

Rejected fits. A fit that failed numerical acceptance is refused by `logLik()`, and through it by `AIC()` and `BIC()`, just as `gt_reliability` refuses it. The two apply one rule: a Gaussian fit is refused on an explicit failure, and a discrete fit must carry an affirmative acceptance, so a discrete object whose acceptance flag is missing or `NA` is refused rather than read as accepted. Its objective is retained in `minus2loglik` for diagnosis, but a likelihood that did not pass the fit's own numerical checks is not a basis for model selection.

Formula and update. `formula()` gives every fit a formula, so `update()` can edit the model of a fit declared either way: the recorded call is re-evaluated in the caller's environment with the formula under `formula`, the previous `outcomes` and `design` removed, and the object of measurement carried along in `measured` so that an edited formula cannot change the scientific target by re-ordering. A design that declares replicates, a batch, explicit nesting, or a facet absent from every random source cannot be expressed completely as a formula, so such a fit refuses a formula change and must be refit from a new `gt_design()`; arguments other than the formula can still be updated. For these designs, `formula()` displays the requested sources only and does not replace the original declaration. Column names containing backticks or backslashes are preserved as names in reconstructed formulas. A fit stored without its data (`retain = list(data = FALSE)`) needs `data =` supplied to `update()`.

Deviance. `deviance()` returns the objective for every fit, including one that failed numerical acceptance, because the value where the optimizer stopped is what a user reads to see how far a refused fit got. It describes the fit and is not a verdict on it: model selection goes through `logLik()`, `AIC()` and `BIC()`, which refuse such a fit.

Discrete fits. The log likelihood is a first-order Laplace approximation to the marginal log likelihood, labelled as such in the returned object. Comparing two Laplace-approximated values compares two approximations, not two exact likelihoods.

Why `vcov()` refuses. In R, `vcov(fit)` is the covariance matrix of `coef(fit)`, and generic tooling relies on that pairing. The package does not expose that covariance through this method. Its OpenMx path profiles the Gaussian outcome means and currently computes uncertainty only for source covariances; profiling does not make mean uncertainty mathematically unavailable. The retained lme4 model can provide its own fixed-effect covariance, but that engine-specific extractor has not been integrated into this package method. Optional discrete observed information includes location and covariance parameters in stored parameter coordinates, unaffected by optimizer scaling. In particular, ordinal log-gap coordinates differ from the natural thresholds returned by `coef()`, and a matching location-covariance extractor is not implemented. Returning a different matrix under that name would be surprising however carefully it were documented, so `vcov()` raises an error naming the reason and pointing at `gt_component_vcov()`.

`gt_component_vcov()` returns available sampling covariance of the source covariance entries, which `gt_reliability` propagates into coefficient intervals. OpenMx Gaussian fits can provide it under their curvature checks. Discrete fits require explicit `check_hessian = TRUE`, numerical acceptance, stable finite differences and usable positive-definite information; a source covariance on or near its boundary is held fixed at its fitted value and the information is formed over the remaining parameters. The default discrete fit and the lme4 Gaussian path provide no such matrix. When unavailable the accessor raises an error carrying the fit's recorded reason, rather than inventing uncertainty. For discrete fits, the `approximation` and `hessian_scale` attributes identify the first-order Laplace qualification and negative-log-likelihood scale. These do not claim finite-sample coverage.

With `type = "parameters"`, Gaussian covariance is $2H^{-1}$ for a Hessian on the minus-twice-log-likelihood scale. Discrete covariance is H^{-1} for the full negative-log-likelihood Hessian, including nuisance intercepts and threshold coordinates. The component matrix is obtained from that full

inverse through the covariance-parameter Jacobian; nuisance location parameters are not fixed during extraction. Specified fixed discrete source covariances correctly have zero sampling covariance conditional on those assumptions, and `conditional_on_fixed` names them. These zero entries do not represent estimated components with no uncertainty. Gaussian and discrete intervals may also condition on boundary components held fixed, which then have zero sampling covariance; `conditional_on_zero` names the components fixed at zero by that boundary policy, while a singular but nonzero source remains fixed at its fitted covariance.

`coef()` returns location parameters on their own scale: Gaussian profiled means; binary intercepts; ordinal thresholds, which are that model's location parameters, with the structurally zero dimension mean omitted; and one intercept per non-reference contrast for unordered categorical outcomes. Variance components are not coefficients; use `gt_components`.

Value

`formula()` returns the model formula: the one the fit was declared with, or for a fit declared through `gt_design()` the source formula built from its outcomes and requested sources. Additional declaration metadata, such as facets absent from all sources, is not represented in that formula. `update()` returns the refitted `gt_fit`, or the call when `evaluate = FALSE`. `logLik()` returns a `logLik` object carrying `df`, `nobs`, a `REML` flag, and a `likelihood` label. `nobs()` returns the number of measurement rows as an integer. `deviance()` returns minus twice the fitted objective as a plain number: the restricted log likelihood under REML, the profile log likelihood under ML, and the first-order Laplace approximation for discrete outcomes. `coef()` returns a named numeric vector of fixed location or contrast estimates. `gt_component_vcov()` returns a labelled covariance matrix with `scope`, `method`, and `conditional_on_zero` attributes, or raises an error explaining why none exists. `vcov()` always raises an error; see below.

See Also

`gt_fit`, `gt_components`, `gt_diagnostics`

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(retry_seed = 42)))
logLik(fit)
deviance(fit)
nobs(fit)
formula(fit)
# Refit the crossed model from the recorded call; at one observation per
# cell the item:rater source is combined with the residual.
crossed <- update(fit, . ~ . + item:rater)
crossed$design
coef(fit)
AIC(fit)   # restricted-likelihood criterion for a REML fit
# The covariance of the estimated source covariances. vcov() deliberately
# refuses: this package does not estimate the covariance of coef().
dimnames(gt_component_vcov(fit))[[1L]]
```

`gt_laplace_check` *Refined Numerical Comparison for a Discrete Fit's Laplace Objective*

Description

Estimates the marginal likelihood of a binary, ordinal or categorical fit at fixed fitted parameters by a numerical comparison method. Reports the difference from the first-order Laplace objective, numerical diagnostics, and whether an empirical stability screen passed. Passing that screen is not an accuracy guarantee.

Usage

```
gt_laplace_check(fit, method = c("auto", "quadrature", "importance"),
  nodes = 20L, draws = 4000L, seed = NULL, max_dimension = 60L,
  max_evaluations = 1000000L, data = NULL)
## S3 method for class 'gt_laplace_check'
print(x, ..., digits = 6L)
```

Arguments

`fit`	A discrete `gt_fit` object. A Gaussian fit is refused: its likelihood is exact.
`method`	`"quadrature"` is adaptive Gauss-Hermite quadrature on a product grid, available up to four random-effect dimensions. `"importance"` uses a defensive mixture: 90% the Gaussian at the conditional mode and 10% the standard-normal prior on standardized random effects. Every importance weight uses the full mixture density. `"auto"` chooses quadrature up to three dimensions and importance sampling otherwise.
`nodes`	Quadrature nodes per dimension, from 5 to 80. The estimate is also computed with five more nodes, and the change is reported.
`draws`	Importance-sampling draws, from 100 to one million.
`seed`	Integer seed for importance sampling, required there so that the check can be reproduced. The caller's random-number state is restored.
`max_dimension`	The largest random-effect dimension the check accepts. Importance sampling can have poor relative precision in large dimensions. Raising this cap or `draws` does not ensure accuracy; examine independent seeds and increasing budgets.
`max_evaluations`	The largest number of integrand evaluations the check will run. Quadrature in `d` dimensions needs `nodes` to the power `d` evaluations plus the same with five more nodes; importance sampling needs `draws`. Both this and `max_dimension` are applied before dense matrices are allocated or solved; the dense-memory budget described below also applies.
`data`	The modelled data frame, needed only when the fit was stored without its data. A fitted Call source is reconstructed from retained item-to-batch membership, so auxiliary call-ID and order columns are optional. The original items and complete fixed layout must be present. Supplied auxiliary call columns must agree with the retained layout.

x	A gt_laplace_check object.
digits	Significant digits for the printed likelihood values.
...	Further print arguments; currently unused.

Details

The Laplace objective is $f(\hat{u})+\hat{u}'\hat{u}/2+\log\det R$, where f is the conditional negative log likelihood of the standardized random effects u, $\hat{u}$ is the conditional mode, and $R'R$ is the Hessian of $f(u) + u'u/2$ at the mode. The marginal likelihood it approximates is $\int \exp(-f(u))\,\phi(u)\,du$. Quadrature adapts a product grid using $N(\hat{u}, (R'R)^{-1})$.

Importance sampling instead uses $q(u) = 0.9\,\phi_H(u - \hat{u}) + 0.1\,\phi(u)$, where ϕ_H has covariance $(R'R)^{-1}$. An independent component indicator selects the sampling distribution for each draw; the denominator of every weight is the full mixture density $q(u)$, not the density of the selected component. For these discrete observation models, $0 \le \exp(-f(u)) \le 1$. Since $q(u) \ge 0.1\phi(u)$, the unnormalised weights $\exp(-f(u))\phi(u)/q(u)$ are bounded by 10 and have finite moments. A Gaussian-only proposal need not have finite weight variance; the prior component addresses that defect without changing the fitted model or Laplace objective.

The difference estimates approximation error in the objective at fixed parameters. It is not a measure of bias in fitted estimates, which would need refitting under a different likelihood. A stable quadrature refinement is evidence of numerical stability, not an error bound. Finite importance-weight moments do not guarantee useful relative precision or calibrated finite-sample Monte Carlo intervals. The log-likelihood estimate has finite-sample Monte Carlo bias, its reported standard error is a delta-method estimate, and empirical ESS can miss poorly sampled regions. Examine independent seeds and increasing draw budgets, preferably against an independent numerical reference on a tractable model. Neither a larger budget nor a passed screen establishes adequacy on another fit. The versioned 'validation-studies/laplace-refinement' record compares the defensive-mixture diagnostic with independent integration on a bounded binary fixture; it is not a general validation of all designs or outcome families.

The comparison introduces its own linear algebra, and that is checked as the engines check theirs: the Cholesky factor at the mode must solve the package's fixed probe systems, every triangular solve that places an integration point is verified against its own system, and an integrand value the response kernel cannot evaluate is counted as unavailable rather than taken as zero. When any of these fails, the result says the refined estimate is unavailable and gives no number in its place.

The diagnostic uses dense conditional algebra even when the fitted engine was sparse. Before allocating dense matrices, it applies the fit's discrete max_dense_bytes control to a conservative estimate of dense reconstruction plus importance-proposal or quadrature-grid workspace. A fit that is feasible with sparse algebra can therefore exceed the diagnostic's memory budget. The dimension, evaluation and draw limits remain in force; raising a memory budget does not improve approximation accuracy. Fixed-layout Call sources use the retained item membership and the same layout validator as fitting. An absent or inconsistent retained layout is refused rather than replaced by newly inferred batches.

The diagnostic reconstructs covariance parameters using the coordinates recorded by the fit. In particular, a saved joint fit using log-Cholesky coordinates is not reinterpreted under a newer Cholesky default. An unsupported or inconsistent recorded parameterization is refused.

This function does not refit the model. Adaptive quadrature as a fitting method in **lme4** and **ordinal** is restricted to a single scalar random-effect term; it is not a direct substitute in those packages for

a crossed multi-source fit. Other Monte Carlo likelihood methods exist for some crossed discrete models. A comparison must match the response family, link, covariance structure and estimand; this diagnostic does not replace such a refitted comparison.

A fit that failed numerical acceptance can still be checked. The result then describes its stopping point, and the print says so.

Value

A `gt_laplace_check` list. `available` says whether a numerical comparison could be computed, and `reason` explains any failure: no Cholesky factor at the mode, a factor that failed the package's solve check, or integrand points that could not be evaluated; `unavailable_evaluations` counts those points. When available, `usable` and `reason` give the verdict of an empirical stability screen: for quadrature, five more nodes per dimension reproduce the estimate to one part in a million of the objective; for importance sampling, empirical effective sample size is at least 200 and at least a tenth of the draws. This is a diagnostic screen, not a certificate of accuracy.

The returned numerical fields are:

- `laplace_minus2loglik`: the objective recomputed at the fitted parameters.
- `refined_minus2loglik`: the numerical comparison estimate.
- `difference`: Laplace minus comparison. Its value per observation is `per_observation`.
- `method`, `random_dimension`, and `observations`: the method and problem size.
- `nodes`, or `draws` with `seed`: the integration controls. The evaluation count is `evaluations`.
- `mc_se`: a delta-method estimate of Monte Carlo standard error in minus-two-log-likelihood units, for importance sampling.
- `effective_sample_size`: the empirical importance-sampling ESS.
- `proposal` and `prior_weight`: the mixture specification.
- `proposal_draws`: the realized component counts.
- `weight_upper_bound`: the theoretical bound on unnormalised weights.
- `max_importance_weight`: the largest observed unnormalised weight.
- `refinement_difference`: the quadrature change on the finer grid.
- `stored_minus2loglik`: the objective retained by the fit.
- `recomputed_agrees`: whether the recomputed objective matches the stored value.
- `numerically_accepted`: the fit's numerical acceptance flag.
- `resources`: estimated bytes for dense reconstruction, integration workspace, their total, and the applied `max_dense_bytes` budget. These conservative estimates are not measured peak memory.
- `scope`: the qualifications on interpreting this comparison.

`print()` displays those qualifications and returns the object invisibly.

See Also

`gt_fit`, `gt_diagnostics`

Examples

```
b <- expand.grid(item = 1:3, rater = 1:8)
b$y <- as.integer((b$item + b$rater) %% 3 > 0)
fit <- gt_fit(b, "y", gt_design("item", "rater", random = ~ item),
  gt_family("binary", "logit"))
gt_laplace_check(fit)
gt_laplace_check(fit, method = "importance", draws = 2000, seed = 1)
```

`gt_pilot_plan` *Evaluate Pilot Precision or Categorical Point-Estimator Performance*

Description

Simulates and refits proposed complete pilot panels. Records interval precision or the performance of point estimates for one fixed target annotation protocol. Increasing pilot item count changes estimation information; changing the target allocation changes the coefficient being estimated. These two choices remain explicit.

Usage

```
gt_pilot_plan(fit, grid, nsim, seed, coefficient = "Erho2", outcome = NULL,
  score = NULL, kind = "outcome", width_target = NULL, level = 0.95,
  target_counts = NULL, components = NULL, means = NULL,
  control = fit$control, max_rows = 1000000L, scale = NULL,
  integration = list())
## S3 method for class 'gt_pilot_plan'
print(x, ..., digits = 4L, max_rows = 12L)
## S3 method for class 'gt_pilot_plan'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`fit`	An accepted fit supported by `gt_simulate`. Interval precision for Gaussian, binary or ordinal outcomes additionally requires `gt_reliability`. Categorical point targets require `gt_categorical_target`, including nominal and modelled fixed-call fits. Joint categorical models retain their complete joint law while reporting one selected outcome.
`grid`	A numeric data frame with one proposed pilot per row. The required `n_items` column gives the object count. Optional facet columns give the pilot's facet counts. All counts must be integers of at least two. At most 100 rows are accepted. Unspecified facets retain the original counts. A facet called `n_items` must first be renamed to avoid ambiguity. Modelled calls require the original item count and group membership in every pilot; only instrumentation counts may change.
`nsim`	Number of attempted replicates per candidate. The total across all candidates must not exceed 10000. Small values are software illustrations, not precise planning evidence.

`seed` Required integer seed. Each attempted replicate receives a recorded deterministic data seed and a separate `retry_seed` for its refit. An explicitly supplied Gaussian retry seed is preserved. Otherwise a separate seed is assigned and recorded for each refit. A categorical refit uses its recorded `retry_seed` as an isolated RNG scope without changing the engine's deterministic starts or numerical acceptance controls.

Point-target pilots additionally receive an independent `integration_seed` per attempt and a separate recorded `reference_seed` for the generating scenario. All these seeds are derived from the master seed before refitting. The caller's random-number state and kind are restored.

`coefficient` Choose the quantity to evaluate:

- `"Erho2"` or `"Phi"`: interval precision.
- `"observed_score"`: observed-score point estimates.
- `"category_vector"`: category-vector point estimates.
- `"decision_agreement"`: decision-agreement point estimates.

The last three use the corresponding targets in `gt_categorical_target` and supply no parameter intervals.

`outcome` One reported outcome. Required for multivariate outcome results.

`score` Optional explicit composite weights from `gt_score` for interval precision. Must be `NULL` for categorical point targets; supply category scores as the `scores` entry of `integration`.

`kind` `"outcome"` or `"composite"`; select one coefficient. Categorical point-target pilots require `"outcome"`.

`width_target` Optional desired total interval width in $(0, 1]$. A precision success requires an available interval no wider than this value. Must be `NULL` for categorical point targets; integration error is not used to construct a parameter confidence interval.

`level` Interval confidence level passed unchanged to `gt_reliability`. Unused by categorical point targets.

`target_counts` Named positive integer instrumentation counts defining the eventual annotation protocol whose coefficient is estimated in every replicate. Unspecified counts, including all counts when `NULL`, use the original fit's counts. They do not change with the pilot candidate. Counts of one are allowed here even though fitting a pilot needs at least two levels.

`components`, `means` Optional parameter scenario overrides as documented for `gt_simulate`.

`control` Refitting controls from `gt_control`; default is the original fit's controls. No numerical acceptance rule or limit is relaxed. Categorical intervals require a fit with available full observed information; request `discrete = list(check_hessian = TRUE)` when appropriate. Information failures remain explicitly unavailable; a refit with a source on its boundary has conditional intervals, flagged as such. Any supplied fixed-covariance controls remain fixed during refitting.

`max_rows` Maximum generated rows per replicate, as for `gt_simulate`, and in the print method the number of candidate pilots shown.

`x` A `gt_pilot_plan` object.

`scale` Must be explicitly `"latent"` for binary or ordinal interval precision. Gaussian fits use `NULL` or `"observed"`. The latent target is not reliability of observed category scores or repeated label decisions. Categorical point targets require `"observed"` for observed-score or category-vector ratios and `"decision"` for decision agreement.

`integration` For categorical point targets only, a uniquely named list: `n_items` (default 200, at least two), `replications` (default 40), `reference_replications` (default 400), `max_rows` (default 100000), and optional `scores`. Both replication counts must be between 2 and 10000. These defaults are computation settings, not a precision guarantee. Scores follow `gt_categorical_target` and must be explicit for ordinal observed-score targets. Integration seeds are generated from the master seed and must not be supplied here. With fixed calls, `n_items` defaults to and must equal the original item count. Instrumentation counts for both reference and refitted targets are always `target_counts`. Leave this list empty for Gaussian or latent interval precision.

`digits` Significant digits in the printed table.

`row.names, optional`
As for `as.data.frame`; `row.names` replaces the row names of the returned data frame and `optional` is ignored.

`...` Further method arguments; currently unused.

Details

The original estimator and resolved source covariance structures are preserved on every refit. Fresh object effects and instrumentation levels are sampled under the declared source law. The scope uses fully random facets without reuse of specific fixed evaluator realizations. Interval-precision pilots refuse batch declarations. Gaussian pilots retain one outcome vector per complete coded cell. Binary and ordinal pilots retain their fitted links, thresholds, joint source covariance and declared equal cell replication. They target explicitly selected latent-response coefficients when `coefficient` is `"Erho2"` or `"Phi"`. The original fitted thresholds define simulation; thresholds and intercepts are reestimated in each refit. Parameter overrides change the generating scenario, not the specified refitting constraints. No missing values are imputed.

Categorical pilot planning is additionally limited to one million generated rows across all attempted refits. Per-fit numerical and workspace limits still apply; this is not a runtime or acceptance guarantee. Samples with absent categories retain all declared levels and are recorded as fitting errors, without recoding or discarding the attempt. Interval pilots also repeat the full information calculation for eligible accepted refits. Its likelihood-evaluation cost grows quadratically with the number of stored parameters; see `gt_control`. A feasible large-panel point fit does not qualify interval-pilot computation at the same size.

Categorical point targets. Observed-score, category-vector and decision targets evaluate the finite numerical estimator rather than interval width. First, the specified generating scenario, including parameter overrides, is integrated independently to obtain one fixed reference estimate. It is a numerical reference, not exact truth; its integration standard error is retained. An undefined or unavailable reference stops the calculation before any refits. Each attempted pilot is then simulated, refitted and evaluated at the same `target_counts`, with a separate inner integration seed. Nominal categories retain their labels and reference contrasts. Fixed calls retain item count, membership and

batch size in both pilot and target panels, while resampling call effects. Variable or newly assigned groups are unsupported.

Available-only bias and RMSE can be affected by numerical-target selection near weak signals. Interpret them together with all-attempt target availability and reference integration error; a missing target must not be replaced by a new seed. See gt_categorical_target for precision limitations.

This route limits all generated rows, including the reference, every pilot, and every planned inner integration, to ten million before calculation starts. The separate one-million pilot-row limit also applies. Reference and inner integrations are both preflighted against the 512 MiB estimated target working-storage limit, including moment matrices. Only the reference estimate and integration metadata are retained during refitting, so its moment matrix does not accumulate with the inner integration workspace. Failed attempts do not cause replacement attempts. Samples from a fit with omitted whole cells are complete future panels under that fitted model and its ignorability assumptions; the procedure does not simulate an unprovided missingness mechanism.

Every planned replicate has a record, including simulation errors, preparation or fitting errors, numerical refusals, reliability errors, and unavailable intervals. Warnings are retained in that record. Error, refusal, and unavailability are not silently deleted from rate denominators. Width summaries condition on an available interval and show their denominator. Precision success rates use all attempted replicates, so failures and unavailable intervals are non-successes; they do not imply a true coefficient or width of zero.

Reported Monte Carlo standard errors for rates use $\sqrt{p(1-p)/B}$, where B is the number of attempted replicates for unconditional rates. The explicitly conditional precision rate and its Monte Carlo standard error instead use the number of available intervals. The mean-width Monte Carlo standard error uses the sample standard deviation divided by the square root of the number of available intervals. A zero plug-in rate standard error after no failures or all failures is not evidence that the underlying probability is known exactly. Increase replicate counts and examine parameter scenarios before making a collection decision.

Intervals retain the fitting engine's asymptotic delta-method interpretation. Conditional boundary intervals are explicitly flagged. This procedure evaluates model-based plug-in precision, not calibrated coverage, a parametric-bootstrap confidence interval, or guaranteed performance in the actual annotation study. It does not select an optimal pilot automatically. Categorical intervals inherit the fitted Laplace approximation. Their precision does not establish approximation accuracy or interval coverage. Fixed covariance components contribute no estimated variance uncertainty; this conditional scenario must not be interpreted as the precision of estimating those components.

For point targets, bias and RMSE compare available estimates with the recorded reference and use only the available-target denominator. Unavailable estimates are not zeros. Empirical standard deviation includes both estimation variation and inner integration noise; the latter is not subtracted automatically. Bias and RMSE Monte Carlo standard errors condition on the recorded reference, whose separate MCSE is shared across every comparison. For A available estimates, the bias MCSE is their sample standard deviation divided by $\sqrt{A}$; the RMSE MCSE uses the sample standard deviation of squared errors divided by $2\,\mathrm{RMSE}\sqrt{A}$. These two standard errors require at least two available estimates, and the RMSE MCSE additionally requires positive RMSE. Neither is a confidence interval for the fitted target or evidence of unconditional accuracy after numerical selection.

Value

A list with `allocations`, one row per proposed pilot; `replicates`, one row per attempted simulation/refit; `summary`, rates, denominators, Monte Carlo standard errors, and conditional width summaries; `scenario`, the parameters, target protocol, estimator, coefficient, scale, seeds, and controls; and `interpretation`. Categorical scenarios additionally retain families, thresholds and the source covariance structure. The object has class `gt_pilot_plan`: `print()` shows each candidate's allocation beside the applicable precision or point-target summaries, and `as.data.frame()` returns one row per candidate, joining `allocations` with `summary`.

`accepted` counts accepted fits even when coefficient extraction fails. `interval_unavailable` counts successful coefficient extraction without a finite interval. `reliability_errors` counts extraction failures separately. The `interval_available` count is the denominator of conditional width summaries. `conditional_intervals` reports how many available intervals hold boundary components fixed. The complete replicate table is authoritative for further stratified summaries.

Point-target results additionally contain:

- `reference`: a one-row target table.
- `reference_integration`: its integration settings and seed.

The replicate table adds:

- `integration_seed`: the inner integration seed.
- `target_available`: target availability.
- `target_raw_estimate`: the untruncated point estimate.
- `target_mc_se`: its integration MCSE.
- `error_vs_reference`: the estimate minus the reference.

Interval columns remain unavailable.

The summary reports acceptance and availability rates with MCSEs using all attempts, together with these target counts and reference fields:

- `target_available`: the number of available target estimates.
- `target_unavailable`: unavailable target calculations.
- `target_errors`: target-integration errors.
- `reference_estimate`: the recorded reference estimate.
- `reference_mc_se`: the reference integration MCSE.

Conditional summaries use the following fields:

- `mean_estimate_given_available`: mean available estimate.
- `bias_vs_reference_given_available`: bias relative to the reference.
- `rmse_vs_reference_given_available`: RMSE relative to the reference.
- `empirical_sd_given_available`: empirical standard deviation.
- `mean_inner_mc_se_given_available`: mean inner integration MCSE.
- `bias_mcse_given_reference`: conditional bias MCSE.
- `rmse_mcse_given_reference`: conditional RMSE MCSE.

Counts and rates distinguish simulation errors, fitting errors, numerical refusals, target-integration errors and unavailable target calculations. No precision-success or coverage claim is supplied for this route.

Two character columns retain diagnostics from each refit. The `acceptance_failures` column contains failure codes separated by `|`. The `selected_attempt` column identifies the returned optimizer attempt. These columns survive ordinary CSV export; whole fit objects are not retained. A missing code or identifier is `NA`, including when no fit was returned. Accepted fits usually have no failure codes, while their selected attempt is retained when recorded. Reliability errors and unavailable intervals retain diagnostics from the completed refit. Missing diagnostics do not imply acceptance; the `accepted` column remains authoritative.

To replay a replicate, generate data with its recorded `seed`, copy `scenario$control`, replace the copied Gaussian `retry_seed` with the recorded refit seed, and refit with the recorded estimator and covariance structures. For categorical replay, use the retained families and source covariance structure and run the refit under its recorded `retry_seed`; there is no Gaussian retry setting to insert. Earlier replicate failures or optimizer retries do not alter later replicates' random streams. For a point target, evaluate the replayed fit with the recorded `integration_seed`, fixed `target_counts`, outcome, category scores and integration settings. The reference uses the generating parameters, `scenario$reference_seed`, and `reference_replications`.

See Also

gt_simulate, gt_reliability, gt_categorical_target, gt_dstudy

Examples

```
# Small synthetic panel and four refits to demonstrate the interface only.
d <- expand.grid(item = 1:12, evaluator = 1:3)
d$quality <- sin(d$item) + d$evaluator / 5 +
  cos(d$item * d$evaluator) / 4
fit <- gt_fit(quality ~ item + evaluator, d, measured = "item",
  control = gt_control(gaussian = list(retry_seed = 42)))
# Each pilot estimates the same three-evaluator protocol.
plan <- gt_pilot_plan(fit,
  grid = data.frame(n_items = c(8, 12), evaluator = c(2, 3)),
  nsim = 2, seed = 2026, coefficient = "Phi", width_target = 0.15,
  target_counts = c(evaluator = 3))
plan
as.data.frame(plan)
plan$replicates
# Two simulations per candidate cannot support a pilot-size recommendation.
```

`gt_plan` *Rank a Bounded Set of D-Study Allocations by User-Supplied Cost*

Description

Exhaustively evaluates a declared set of balanced facet counts under one fitted model. It retains every candidate, screens a reliability target, identifies minimum-cost ties and a cost–performance Pareto set, and explains changes in error contributions relative to a declared baseline candidate.

Usage

```
gt_plan(fit, candidates, cost, currency, cost_date, study_items, target,
        coefficient = "Erho2", outcome = NULL, kind = "outcome",
        screening = c("point", "lower"), baseline = 1L,
        constraints = NULL, scale = NULL, score = NULL,
        fixed = character(), level = 0.95, max_candidates = 10000L,
        cost_description = NULL, integration = list())

## S3 method for class 'gt_plan'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'gt_plan'
print(x, ..., digits = 4L, max_rows = 12L)
```

Arguments

`fit`	An accepted `gt_fit` supporting analytic balanced reliability, or a categorical fit supporting `gt_categorical_target` for an explicit observed or decision target. Modelled categorical calls retain their fitted item count and layout; analytic planning still refuses call effects.
`candidates`	A nonempty numeric data frame of instrumentation facet counts, or a named list of count vectors expanded to their full Cartesian product. Omitted facets retain their fitted counts. Counts must be positive integers below 2^{53}. Duplicate rows remain distinct candidates.
`cost`	A numeric data frame of nonnegative, finite monetary cost components, with one row per candidate, or a function called once with a planning data frame. The planning frame contains `design_id`, `allocation_` prefixed facet counts, `annotations_per_item`, `study_items`, and `total_annotations`. It has an `allocation_columns` attribute mapping column names to the original facets. The function returns a cost-component data frame or a list containing `costs`. Such a list can also be supplied directly. Its optional entries are: • `resources`: a nonnegative numeric data frame. • `assumptions`: character text. Components are summed: do not include an already summed total as an extra component. Prices, usage assumptions, overhead, retry costs, evaluator setup and human-review costs are entirely user supplied.
`currency`	One three-letter uppercase currency code; all cost components must already use that same currency. No exchange conversion is performed.
`cost_date`	The date of the supplied cost assumptions, as `Date` or `"YYYY-MM-DD"` text.
`study_items`	The number of distinct items to be costed. This is a positive integer held constant across candidates. It multiplies annotations and costs; it does not change reliability coefficients or estimate pilot-study precision.
`target`	A number from zero to one for the selected coefficient.
`coefficient`	The coefficient or repeated-panel target to screen: • `"Erho2"`: relative comparisons.

- "Phi": absolute decisions.
- "observed_score": categorical score target.
- "category_vector": category-proportion target.
- "decision_agreement": plurality-decision agreement.

The last three use gt_categorical_target. Its corresponding target names are "score", "category_vector" and "decision", respectively.

outcome — The outcome to screen. Required when multiple outcomes are reported.

kind — "outcome" or "composite"; a composite requires score. Categorical target planning permits only "outcome".

screening — "point" screens the projected point estimate. "lower" separately screens its available lower confidence bound at level. Missing bounds remain unavailable and do not pass the screen. Categorical observed and decision targets permit point screening only.

baseline — The one-based candidate row identifying the comparison protocol; default is row 1. Its complete allocation and cost are retained. The baseline can be infeasible: it remains a comparison, not a recommendation.

constraints — Optional function applied to the joined candidate, coefficient, cost and resource table, or a row-aligned logical vector/data frame supplied directly. A function returns a logical vector or a data frame of named logical checks. FALSE excludes a candidate; NA records unknown feasibility. The checks and their reasons are retained. Cost components use the prefix cost_component_; resource columns use resource_.

scale, score, fixed, level — As in gt_reliability. The score and fixed-facet universe are declared once for all candidates. A fixed facet must retain its observed count; it is never selected or changed by the optimizer.

max_candidates — Positive integer upper limit checked before Cartesian expansion; default 10,000. The caller defines the search bounds.

cost_description — Optional explanatory text recording the source and units of the cost model. Supply additional assumptions through the cost record.

integration — For categorical target planning only, a uniquely named list with an explicitly supplied seed; optional n_items (default 200), replications (default 40 independent panel pairs, range 2–10,000), max_rows (default 100,000 per panel, maximum 1,000,000), and scores. For a modelled call layout, n_items defaults to and must equal the fitted item count, as must study_items. Category scores follow gt_categorical_target and must be explicit for ordinal outcomes. Use scale = "observed" for score/vector targets or scale = "decision" for decision agreement. Composite score objects and fixed facets are not supported on this route. Every candidate facet count must be at least one. All candidates together are limited to ten million generated rows, counting both panels in every replication; this guard is checked before simulation begins.

x — A gt_plan object.

row.names, optional — Standard data-frame conversion arguments.

`...` Reserved for S3 compatibility.

`digits`, `max_rows`
Printing precision and maximum displayed rows.

Details

Three distinct sizes. `study_items` counts distinct study items. `annotations_per_item` is the product of facet counts and the fitted equal within-cell replication. `total_annotations` is their product. Items per request, request counts and retry counts may be supplied as cost resources; they are not statistically interchangeable with repetitions per item. Total annotations must remain exactly representable integers below 2^{53}.

What is optimized. All allocations use the same accepted fit, source covariances, score, outcome scale and declared facet universe. Counts represent exchangeable levels under that model. The function does not replace an expensive evaluator with a cheaper identity, refit an evaluator subset, change a random facet into a fixed facet, or alter a batching layout to improve a coefficient. A heterogeneous cost model must still describe costs under that same statistical sampling universe. Evaluator-identity selection needs separate model evidence. Adding a human-review cost does not by itself model a reliability benefit.

Search and missing values. A candidate is eligible when all constraints are satisfied and the selected screening value reaches the target. Minimum-cost flags retain every exact cost tie among eligible candidates. Pareto status compares cost with the selected performance measure among feasible candidates having an available screening value, independently of the target. A design is dominated when another is no more expensive and performs at least as well, with at least one strict improvement. Exact duplicate cost–performance ties remain non-dominated. Unknown feasibility or performance has `NA` Pareto status; known infeasible rows have `FALSE`. An unavailable lower bound is not replaced by its point estimate or a zero standard error. A missing lower bound cannot qualify even if the point estimate exceeds the target.

Interpretation of intervals. The available Gaussian intervals retain the parent reliability calculation's model and any conditional boundary qualification. They are pointwise estimates of uncertainty, not simultaneous coverage guarantees after searching many designs and not future-panel prediction bounds. Discrete latent intervals require an available observed-information calculation; unavailable intervals keep their reason. For categorical observed and decision targets, `mc_se` measures numerical integration error at fixed fitted parameters. It is never substituted for parameter uncertainty, a lower confidence bound, or uncertainty after selecting a design.

Batching boundary. Analytic planning refuses modelled `Call` effects. Categorical target planning simulates the fitted equal fixed layout, retaining the item-to-batch membership and fitted item count while allowing new exchangeable facet levels. It does not optimize batch size, item grouping, or ordering effects. Request batching in a cost callback does not itself introduce shared-call dependence into a fit without a call source.

Categorical targets. The selected outcome and category scoring remain fixed across candidates; all dimensions of a joint fit are still simulated. The planner uses the same explicit seed for each candidate, which preserves duplicate allocations exactly. This does not supply paired-comparison inference between candidates. Minimum-cost and Pareto results rank Monte Carlo point estimates, so small changes can reflect integration error. Increase integration precision when that affects a decision. Decision agreement is repeatability of the declared vote rule, not discrimination or correctness; its decision-diversity diagnostic is retained beside agreement. Before integration starts, a 512 MiB estimated working-storage guard accounts for all retained candidate moment matrices plus

the largest active integration workspace. The moments remain available in `target_projections`; the estimate is recorded as `integration_resource_budget` in `provenance`.

Marginal explanations. Changes are candidate minus baseline for cost and coefficients, and baseline minus candidate for relative/absolute error variance. Negative reductions therefore identify increased error. Per-source reductions add to their corresponding total error reductions. These are conditional projections, not causal effects, empirical accuracy, or a promise that future estimates will attain the target. Categorical target planning reports `target_gain` relative to the baseline, without a Gaussian source-error decomposition or invented G/Φ columns.

Value

A `gt_plan` list. `results` (also `as.data.frame(x)`) retains all candidate allocations, coefficients and intervals, cost components/resources, constraint status/reasons, target status, minimum-cost and Pareto indicators, cost rank among feasible rows, and baseline cost/error/coefficient changes.

`source_contributions` reports universe, relative-error and absolute-error contributions and the error reductions per source. For a nominal outcome the scalar decomposition is unavailable. Its coefficient is the mean generalized eigenvalue over its contrast dimensions, which has no single variance decomposition, so its source contributions are `NA`, as are its `universe_variance` column, its `relative_error` column and its `absolute_error` column; the `decomposition` entry of `provenance` says so. Allocations are still screened and costed on the nominal coefficient. The input allocations and full coefficient tables remain in `allocations` and `all_coefficients`. Cost and resource records remain in `cost_breakdown` and `resources`. The `constraint_checks` element preserves every check. The `baseline` element records the chosen comparison.

`provenance` records the currency, date, cost assumptions and callback text. It also records the search scope, target, score, facet universe, interval qualifications, original fitted counts and selected baseline. `fit_diagnostics` retains the fitted numerical diagnostics. No request is executed and no provider price is queried.

On the categorical route, `results` retains the target estimate, `mc_se`, availability and reason, raw ratio and target-specific moments or decision diversity. `target_projections` retains each underlying target object or its failure record; `source_contributions` is empty. Integration settings, aggregate simulation work, and the uncertainty interpretation are retained in `provenance`. Missing candidate targets remain in the table and cannot qualify for minimum cost or Pareto status.

See Also

`gt_reliability`, `gt_dstudy`, `gt_dstudy_target`, `gt_categorical_target`

Examples

```
# Small synthetic Gaussian example; no data or prices are downloaded.
d <- expand.grid(item = 1:12, evaluator = 1:3)
d$quality <- sin(d$item) + d$evaluator / 5 +
  cos(d$item * d$evaluator) / 4
fit <- gt_fit(quality ~ item + evaluator, d, measured = "item",
  control = gt_control(gaussian = list(retry_seed = 42)))
plan <- gt_plan(fit,
  candidates = list(evaluator = 2:4),
  cost = function(d) list(
    costs = data.frame(annotation = 0.01 * d$total_annotations,
```

```
                      setup = 2 * d$allocation_evaluator),
    resources = data.frame(request_count = d$total_annotations),
    assumptions = "Illustrative monetary units; one item per request."),
  currency = "USD", cost_date = "2026-10-09", study_items = 1000,
  target = 0.80, coefficient = "Phi", baseline = 1,
  constraints = function(d) data.frame(
    budget = d$total_cost <= 150,
    requests = d$resource_request_count <= 10000),
  cost_description = "Illustrative costs; not provider prices.")
plan
subset(as.data.frame(plan), minimum_cost %in% TRUE)
plan$source_contributions
```

---

gt_preflight *Inspect Data, Source Dimensions, and Backend Limits Before Fitting*

Description

Reports observed design and outcome information, random-source dimensions, parameter counts, and the current backend's structural and resource restrictions without optimizing a model. Declared absent categories are reported as a failed check; fitting continues to reject them. Other category and source-resolution rules match fitting.

Usage

```
gt_preflight(data, outcomes, design = NULL, family = gt_family("gaussian"),
  covariance = "unstructured", residual = NULL,
  control = gt_control(), max_examples = 10L, formula = NULL,
  measured = NULL)

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

## S3 method for class 'gt_preflight'
plot(x, type = c("cells", "sources", "outcomes"),
  max_sources = 20L, outcome = NULL, max_categories = 12L,
  max_variables = 8L, max_label_chars = 18L, ...)
```

Arguments

data	A nonempty data frame with unique column names. With a model formula it may also be given second, after the formula.
outcomes	One or more outcome column names, or a two-sided model formula as accepted by gt_fit, which then also supplies the design.
design	A design from gt_design. Omit it with a formula; a family given in this position after a formula is read as family.
family	A family from gt_family, or a named family list matching the outcomes.

`covariance` The source covariance specification accepted by `gt_fit`.

`residual` Gaussian residual covariance specification, or `NULL` for the default. Do not specify this argument for discrete outcomes.

`control` Engine controls from `gt_control`. Resource limits are read from the selected engine's controls. Discrete parameter-bound overrides are validated with the same coordinate and representability rules as fitting. Validation of resolved covariance coordinates, starting values and parameter domains runs only after the preceding structural, category and resource checks pass. A request stopped by those earlier checks has not passed the skipped validation; fitting repeats all applicable input checks.

`formula, measured` The model formula under its own name and the right-hand variable that is the object of measurement, as for `gt_fit`.

`max_examples` A nonnegative finite integer, no greater than `.Machine$integer.max`. Maximum rows in each panel-audit table and in each outcome/variable no-variation example table, and maximum invalid design-row indices in an error. Zero suppresses examples while retaining counts and truncation metadata. Defaults to 10.

`x` An object returned by `gt_preflight`.

`type` Plot mutually exclusive full-cell counts (`"cells"`), random-source dimensions (`"sources"`), or descriptive category coverage across design-variable groups (`"outcomes"`).

`max_sources` A positive finite integer limiting the source bars. Sources are sorted by decreasing random dimension, with ties in source-table order. The plot explicitly reports truncation. Ignored for cell bars, but still validated.

`outcome` For `type = "outcomes"`, one outcome name. Required when preflight contains multiple outcomes. Gaussian outcomes have numeric profile tables, not a category heatmap.

`max_categories, max_variables` Positive finite integers limiting heatmap rows and columns. Categories follow declared order; variables follow design order. Displayed and total counts, and truncation, are printed on the plot.

`max_label_chars` A finite integer of at least four, limiting heatmap label text before a position number is added. Abbreviated labels map to the complete profile tables by that number.

`...` For cell/source plotting, named arguments passed to `barplot`; for category coverage, named arguments passed to `image`. Supplied arguments override plotting defaults. Reserved for print-method compatibility.

Details

The panel audit uses the Cartesian product of the levels actually observed in each design column. Unused factor levels are excluded. It does not infer unobserved levels, physical crossing, or permitted combinations from sampling intent. No data, source specification, or model is automatically corrected. `panel_audit` contains:

- summary: one row each for complete, under_replicated, over_replicated, and missing. These categories are mutually exclusive. Complete means an observed full cell has exactly the declared replication; under-replicated cells have at least one row but fewer than declared; missing cells have no rows. cell_count_exact stores exact decimal counts as character strings. cell_count is a numeric convenience column: counts above 2^53 can be rounded, and counts beyond the finite numeric range are Inf. expected_cells_exact gives the exact full-product count.
- missing_cells: representative absent full cells, retaining the original design-column names, sampled values, and classes. Factor levels not present in an example table are dropped, including all levels in empty tables, so factor attributes do not expose omitted identifiers. Examples use first-observed level order, with the object column varying fastest.
- replication_issues: cells with incorrect replication, in input order. Design columns are followed by observed replication, declared replication, and status metadata.
- row_examples: input-row examples from cells with incorrect replication, in input order. Only design columns and audit metadata are copied; outcomes and unrelated data columns are excluded. The row metadata is the one-based input-row position, independent of data-frame row names.
- metadata_columns: maps metadata roles to table column names. The roles identify the input row, observed and declared replication, and status. The corresponding keys are:
  - row
  - observed_replicates
  - declared_replicates
  - status

  Metadata normally uses the prefix .gt_; leading dots are added until none of the metadata names collide with input column names. Original column names are not repaired or renamed.
- examples: for each example table, the requested limit, number shown, total available (total and authoritative total_exact), and whether examples were truncated.

Examples are deterministic, not random samples. Their counts use all input rows. Missing-cell examples are generated incrementally, visiting at most the observed-cell count plus the requested example count; the full Cartesian grid is never materialized. All three example tables are capped separately.

Each outcome_profile[[outcome]] contains:

- family, link, and summary: the number of rows and distinct observed values, and declared/observed category counts for discrete outcomes. Gaussian outcomes instead have minimum, maximum, mean, and sample standard deviation; derived summaries outside the finite numeric range are NA. Gaussian values are not converted to categories. Globally constant, missing, or nonfinite Gaussian inputs still raise the fitting validation error.
- categories: declared category labels, counts, row proportions, and an observed flag. Safely resolved declared categories, including unused declared factor levels, appear with zero counts when absent. The binary numeric/logical category labels follow fitting's 0/1 convention. This table is NULL for Gaussian outcomes.
- by_variable: the number of observed groups, groups with exactly one distinct response (no_variation_groups), groups with one row (single_row_groups), and the no-variation fraction for each design variable. Single-row groups are included in no-variation counts, and counted separately to distinguish missing replication from repeated constant responses.

- `category_coverage`: for each variable/category pair, the number and proportion of observed groups containing at least one row in that category. The denominator is the observed group count, not the measurement-row count. A group may contain several categories, so these proportions need not sum to one. This table is `NULL` for Gaussian outcomes.
- `group_members`: the columns defining each variable's groups. Declared nested variables include all declared ancestors. Other variables use their marginal coded levels. Reused nested IDs are distinguished by their parents. No additional physical nesting is inferred. The `grouping_scope` column of `by_variable` records this choice:
  - `declared_parent_scoped`: groups include declared ancestors.
  - `marginal_coded_levels`: groups use one variable's coded levels.
- `no_variation_examples`: one table per design variable containing at most `max_examples` grouping identifiers, row counts and distinct-value counts, in first-observed order. Sampled values, column classes, and order are preserved; unused factor levels are dropped. Outcome values and unrelated columns are excluded. `metadata_columns` maps `row_count` and `distinct_values` to collision-safe metadata names. `examples` reports total, shown, limit, and truncation for each variable.
- `scope` and `privacy`: interpretation and sharing notes. These tables are descriptive, with no minimum category count or variation threshold for adequacy. They do not certify identification, fit acceptance, or likelihood-approximation accuracy, and do not collapse categories. Category labels, small counts, and sampled grouping identifiers may be sensitive. `max_examples = 0` suppresses group identifiers in example tables, including unused factor levels, but leaves aggregate counts and category/outcome labels. The preflight object is not anonymized.

The recorded call keeps its data argument only when that argument is a plain object name. A data frame supplied through `do.call()`, or an inline expression, is recorded as a marker rather than stored in the call.

For a design made with a `gt_batch` declaration, `batch_audit` contains the declared size, order source, facet and reaches, and `membership`: `"inferred"` when the batches are cut from the item order, `"recorded"` when the calls are read from the column named in the declaration's `id`. It counts the items, the distinct `batches`, the conditions and the `calls`. Recorded calls are counted. Inferred calls are implied: one for each batch, condition and declared replicate. `equal_sized`, `calls_below_size` and `smallest_observed_call` describe calls with fewer items present than declared. That is a property of the rows supplied, not a problem, and not evidence of how many items a call was sent: a call sent short and a call that lost rows after collection look the same, and a call with no row left is not counted. `position_scope` records that positions are ranks among the items present. `fixed_composition` and `fixed_order` say whether every condition holds the same batches, and each batch in one order. `consistent` is `TRUE` when the declaration can be laid over the data, with the reasons in `problems` when it cannot. At most `max_examples` rows list items with their batch, or their recorded call, and position. The object column retains its original name. `batch_audit$metadata_columns` maps the roles `batch` (inferred membership) or `call` (recorded membership), and `position`, to their actual columns in `batch_audit$examples`. These normally use the role names. If an object name collides, metadata names receive a `.gt_` prefix, extended with leading dots if needed; the object name is unchanged. Use this mapping when selecting example metadata by name. `model` says whether a fit of these data would estimate a shared call effect, with the reason when it would not, the name of the source it adds and that source's covariance parameters, which the total parameter count includes. No coefficient scale is offered for a design whose fit models a call effect; see `gt_batch`.

Inferred membership is a reconstruction from the rows supplied, and `consistent` is then no evidence about how the data were collected: items removed after collection, calls of other sizes and regrouping between conditions leave no trace. `membership_scope` says so in the object and the printed report repeats it; for recorded calls it states the limits just described. For inferred batches `stored_rows` reports, when every condition holds each item once, in how many conditions consecutive blocks of rows hold one declared batch, and in how many they also follow the declared order. That agreement means something only when rows are stored call by call. The audit estimates no dependence and does not enter `checks`.

For absent declared categories, the profile and structural/resource estimates remain available; free-covariance and source-kernel checks are skipped after the failed category check, and no reliability scale is offered.

The default plot shows horizontal bars for the four exclusive cell categories. Direct count labels make zero and small counts visible beside large bars; labels above `1e12` use abbreviated scientific notation marked with a tilde. Exact counts remain available in the audit table. The source plot shows the largest random dimensions, up to `max_sources`. Plots describe observed coded levels, not runtime, statistical identification, or fit acceptance. Numeric cell bars may be rounded above `2^53`; a cell plot raises an error when counts exceed the finite plotting range. Consult the exact count strings in that case.

The outcome heatmap shows group coverage for one discrete outcome, with a fixed zero-to-one color scale and bounded category/variable labels. An asterisk marks a variable whose groups include declared nesting ancestors. Small displays (at most eight variables and twelve categories) include percentage labels; near-zero and near-one values are abbreviated without labeling them exactly zero or one. The plot does not show per-group response rows or treat sparse coverage as an automatic model rejection. Larger displays use colors only.

Each random source has one predictor dimension per Gaussian, binary, or ordinal outcome and one dimension per nonreference category for a categorical outcome. Its random dimension is the product of that dimension count and its observed grouping count. These are conceptual random-effect dimensions for Gaussian fitting; the exact Gaussian backend instead uses factorial contrasts.

The source table counts free variance/covariance parameters under the requested covariance structure. Fixed discrete source covariance matrices contribute zero free covariance parameters. Gaussian observation parameters are the profiled outcome means; discrete observation parameters are intercepts or ordinal thresholds. The total includes these observation parameters and the Gaussian residual covariance parameters. It is not a residual degrees-of-freedom count or an information-criterion definition.

Default Gaussian checks require a complete coded Cartesian panel with one measurement per cell and resolve a full-cell source into the residual when appropriate. With `engine = "lme4"`, `panel = "observed"`, missing whole cells are permitted for one Gaussian outcome, subject to the row and memory bounds. Within those bounds, preflight constructs source and residual covariance kernels and checks independence after removing the intercept. The check is limited to algebraic identification and does not assess the missingness mechanism. Batch declarations, replicated cells and joint outcomes are refused on that route. Discrete fitting can also admit missing whole cells. Ordinary analytic reliability still requires a complete balanced coded panel; the observed-row Gaussian route needs a separately requested intended balanced D-study projection. Within configured size limits, preflight applies the current discrete source-kernel checks, including the call kernel for an eligible fixed-layout batch. When an earlier structural or resource check fails, `kernel_check` explicitly reports that the kernel checks were skipped. Source counts refer to the requested model if family-specific resolution failed. With `gradient = "analytic"` in the

discrete control, preflight also reports `analytic_gradient_coordinates`, whether the resolved covariance coordinates support the route, and `analytic_gradient_caps`, whether the design lies within the compiled selected inverse's declared caps, and records the measured sizes and the caps in `resources$analytic_gradient`. With `boundary_escape = "projected"`, preflight reports `boundary_escape_coordinates`, the same coordinate rule the fit applies at start.

The Gaussian allocation estimate covers preparation arrays only, including dense covariance and identification working arrays on the observed-row route. The selected discrete estimate includes random-design, conditional-curvature and factorization arrays and a planning multiplier. The sparse estimate keeps its complete-fill bound when that bound fits the budget; otherwise it refuses on the exact structural nonzero count of the conditional Hessian, or factorizes the structural pattern once and counts its fill, as described in `gt_control`; the counted fill is a planning estimate with the multiplier as allowance, not a bound on the fitted factor. None predicts peak memory or runtime, and none establishes large-panel qualification. Fewer than five observed groups triggers a descriptive note, not a rejection rule. Starting values and optimizer-specific controls are validated by fitting. Optional discrete observed-information calculations occur after fitting, never in preflight.

Passing preflight does not establish statistical identification, reliable estimation, numerical acceptance, or Laplace-approximation adequacy. Supported scales remain conditional on a numerically accepted fit: observed Gaussian scores, or explicitly requested latent binary, ordinal or nominal responses. Latent coefficients describe averages of latent responses, not majority-vote decisions or observed label proportions. A nominal outcome's coefficients are the mean and range of the generalized eigenvalues over its category contrasts, under exchangeable or unstructured covariance; see `gt_reliability`.

Value

A `gt_preflight` list with observed counts and replication, source and parameter totals, resource estimates, and interpretation notes. It also contains:

- `engine`: the engine a fit of this request would use, under the label `gt_diagnostics()` reports: the exact balanced Gaussian engine, the lme4 engine when `control` selects it, or the selected dense or sparse discrete Laplace engine. With lme4 the checks also record whether lme4 is installed and whether the request is univariate.
- `sources`: a data frame describing the random sources a fit would estimate: the retained declared sources and, for a batch the fit would model, the call source. The parameter and dimension totals describe these.
- `random_dimension`: the sum of `sources$random_dimension`, with each source's observed group count multiplied by its predictor dimension count. This includes a modelled call source when present. For Gaussian requests it is a conceptual random-effect dimension, not the dimension of a dense matrix allocated by the exact factorial engine.
- `resources`: engine-specific allocation estimates and limits.

  For complete-panel Gaussian requests, `resources$factorial_strata` is 2^a, where a counts the object and declared facet axes, plus one extra axis when the fit models batches. This describes the exact balanced representation; it is also reported for the lme4 route, whose estimates are checked against that exact deviance. It is absent for observed-row Gaussian and discrete requests. Observed-row Gaussian resources include `preparation_estimated_bytes`, `preparation_limit_bytes` and `max_observed_rows`. When its bounded identification calculation passes, `resources$identification` records the rank, component count, normalized kernel eigenvalues, relative tolerance and scope.

Dense discrete requests report these resource fields:

- `dense_working_bytes_estimate`: estimated working bytes.
- `dense_working_bytes_breakdown`: estimate by operation.
- `dense_matrix_bytes_lower_bound`: matrix-storage lower bound.
- `max_dense_bytes`: requested dense memory limit.

Sparse discrete requests instead report:

- `sparse_working_bytes_estimate`: estimated working bytes.
- `sparse_working_bytes_breakdown`: estimate by operation.
- `sparse_estimate_method`: how the estimate was made. When the complete-fill bound already fits the budget it is reported as `"complete_fill_bound"`. When the conditional Hessian's structural nonzeros alone, counted exactly from the observed group pairs, exceed the budget, the request is refused before any pattern is built and the method is `"pattern_exceeds_budget"`. Otherwise the structural pattern is factorized once and the method is `"structural_factorization"`.
- `sparse_structural_entries`: the exact structural nonzero count of the conditional Hessian and, after a structural factorization, the counted Hessian and factor entries; the two Hessian counts agree.
- `max_sparse_bytes`: requested sparse memory limit.

Both report observation, random-dimension and parameter limits. Only the selected engine's memory bound is applied.

- `panel_audit`: bounded panel diagnostics, described below.
- `outcome_profile`: a named list of descriptive outcome summaries, category counts, grouping coverage, and bounded no-variation examples.
- `batch_audit`: for a design that declares batches, the calls that declaration implies or records; absent otherwise. Described below.
- `checks`: a data frame of the structural and resource checks.
- `fitting_feasible`: whether those checks passed.
- `supported_reliability_scales`: conditional coefficient scales. Complete balanced nominal panels support `"latent"` with `covariance = "exchangeable"` or `"unstructured"`; their coefficients summarize generalized eigenvalues over category contrasts, not observed-label agreement. Diagonal nominal contrasts have no such reference-invariant coefficient. See `gt_reliability`.

Feasibility is not a fitting or identification result. Invalid input types or undeclared observation categories raise an error. Declared but absent categories produce a failed `outcome_category_coverage` check and `fitting_feasible = FALSE`; no category is removed and `gt_fit()` still raises an error. Unsupported observed designs and exceeded resource limits are reported as failed checks. Both print and plot methods return `x` invisibly.

See Also

`gt_fit`, `gt_diagnostics`, `gt_reliability`, `gt_dstudy`

Examples

```
# Entirely synthetic evaluator/prompt data; no API or study data are used.
d <- expand.grid(item = 1:8, evaluator = 1:3, prompt = 1:2)
d$score <- sin(d$item) + d$evaluator / 5 + cos(d$item * d$prompt) / 4
design <- gt_design("item", c("evaluator", "prompt"),
  random = ~ item + evaluator + prompt + item:evaluator)
report <- gt_preflight(d, "score", design)
print(report)
report$sources

# Missing whole cells block the default Gaussian route and ordinary reliability.
incomplete <- gt_preflight(d[-1, ], "score", design)
incomplete$checks[!incomplete$checks$passed, ]
incomplete$panel_audit$missing_cells
plot(incomplete)
plot(report, type = "sources", max_sources = 5L)
if (requireNamespace("lme4", quietly = TRUE)) {
  observed <- gt_preflight(d[-1, ], "score", design,
    control = gt_control(gaussian = list(engine = "lme4", panel = "observed")))
  observed$checks
}
```

`gt_reliability` *Compute Balanced G and Phi Coefficients*

Description

Forms universe-score, relative-error, and absolute-error covariance matrices under a declared balanced averaging design.

Usage

```
gt_reliability(fit, scale = NULL, score = NULL, design = NULL, counts = NULL,
  fixed = character(), level = 0.95, bootstrap = NULL, interval_type = NULL)
## S3 method for class 'gt_reliability'
print(x, ..., digits = 4L, max_rows = 12L)
## S3 method for class 'gt_reliability'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

`fit`	A numerically accepted `gt_fit` with a complete balanced coded panel.
`scale`	`"observed"` for Gaussian outcomes (default); explicitly request `"latent"` for binary, ordinal and nominal outcomes.
`score`	Optional `gt_score` weights. Per-outcome results are always returned.
`counts`	Optional named positive integer instrumentation counts overriding the fitted allocation. Unspecified counts retain their fitted values.

design	Compatibility alias for counts. Supply only one of these arguments.
fixed	Instrumentation facets whose universe of generalization is exactly their observed levels. Defaults to none, which is the fully random model. A fixed facet's count cannot be changed, and at least one facet must remain random.
level	Confidence level for the reported coefficient intervals.
bootstrap	Optional gt_bootstrap result for this fit. Its accepted replicates then supply the standard errors and intervals in place of the delta method; the point estimates remain the fit's.
interval_type	"wald", the default without a bootstrap; "percentile", the default with one; or "bca", which needs the bootstrap's leave-one-out refits.
x	A gt_reliability object.
digits	Significant digits used when printing.
max_rows	Maximum number of outcome rows to print.
row.names	Optional row names for the exported data frame.
optional	Accepted for generic compatibility. Exported column names are always preserved without syntactic name repair.
...	Reserved for additional print or data-frame arguments.

Details

Tabular export. as.data.frame() returns an ordinary data frame with one row per outcome and an additional row for a requested weighted composite. design_id is 1 for this single allocation. kind is "outcome" or "composite"; use it together with outcome, because a measured outcome may itself be named "composite". Coefficients retain the names Erho2 and Phi, with their _se, _lower, and _upper columns, including missing values when uncertainty is unavailable.

Allocation columns prepend allocation_ to each exact facet name. The mapping attribute is named allocation_columns. It maps exported names back to the original facets, protecting coefficient fields from similarly named facets. The table also includes:

- scale and interval_level: coefficient interpretation.
- uncertainty_available: availability under the selected method.
- uncertainty_reason: the recorded reason.
- uncertainty_conditional: the conditional-inference flag.
- uncertainty_conditioning: what was held fixed.
- uncertainty_method: how the standard errors and intervals were obtained, for example the delta method or "object_bootstrap_percentile".
- Erho2_interval_available: relative interval availability.
- Phi_interval_available: absolute interval availability.
- fixed_facets: the declared fixed universe.
- composite_weights: weights for composite rows.
- measurements_per_object: the allocation total.
- measurement_count_exact: integer precision status.

- `extrapolated`: extension beyond fitted facet counts.
- `batch_status`: whether the design declared batches.

`uncertainty_available` describes availability under the method in `uncertainty_method`: for the delta method, a usable fitted parameter covariance; for a bootstrap, enough accepted replicates and, for BCa, a complete set of leave-one-out refits. Individual intervals can still be missing, for example at a coefficient boundary or, for a bootstrap, beyond the replicates' support. The conditional flag records inference restricted to the interior parameter block, and the conditioning text identifies what was held fixed, distinguishing zero from singular nonzero covariance. The per-coefficient availability flags indicate that the estimate and both interval bounds are finite in that particular row. Fixed facets are listed with quoted, escaped labels, separated by semicolons; an empty string means none. Composite rows carry similarly quoted outcome labels and numerical weights; `composite_weights` is `NA` for measured-outcome rows. `measurement_count_exact` is `TRUE` for finite allocation totals below `2^53`, the range in which consecutive integers have exact double representations. Larger totals are retained for reporting but not treated as exact counts when comparing candidate minima. `batch_status` is `"not_declared"`, or `"declared_not_modelled"` for a design made with `gt_batch` whose batches the fit did not model: the coefficients in that row treat items annotated in one call as independent. Printing and plotting say the same. A fit that does model a call effect is refused by these ordinary methods. Use `gt_batch_reliability` and `gt_batch_dstudy` for explicit targets conditional on the fitted fixed Gaussian batching layout, or `gt_categorical_target` for categorical repeated-panel targets under an eligible fixed layout. Neither defines a universal batch-adjusted G.

Columns are atomic vectors suitable for CSV output. Additional attributes retain the fixed-facet declaration, a compact uncertainty record, the structured score, the batch status and the interpretation. Their names are `fixed_facets`, `uncertainty`, `score`, `batch` and `interpretation`. CSV does not preserve R attributes; keep these alongside a CSV when reporting the full analysis context. Conversion does not refit the model or create new uncertainty estimates.

Balanced coded panel. This is the complete Cartesian product of the observed coded levels of the object and of every instrumentation facet, carrying exactly the declared number of rows in each of those cells, with no missing outcome. For a nested facet, *coded* means the child's within-parent code rather than a globally unique label, so the child's identity is the child-by-parent source; `gt_design` works both codings through. A panel that is not balanced in this sense has no analytic coefficient here, and none is approximated.

In the fully random model, the object main-effect covariance defines universe-score variation. Every object-containing error component contributes to relative error, and every non-object-main component contributes to absolute error. Each component is divided by the product of the instrumentation counts in its grouping term; the residual is also divided by declared cell replication.

Declaring a facet `fixed` selects the mixed model of Brennan (2001). The universe of generalization then contains exactly that facet's observed levels: an object interaction containing only fixed instrumentation facets is averaged over those levels and added to universe-score variance; a source built only from fixed instrumentation facets shifts every object equally and leaves the comparison; every remaining source keeps its usual divisor. With no fixed facet this reduces exactly to the fully random decomposition. `$source_roles` records which role each source took. Fixed versus random treatment follows the intended universe of generalization, not a facet name or whether its levels were researcher-selected. The option changes coefficient aggregation after fitting; it does not repair a misspecified G-study covariance model.

Intervals use the delta method on the logit scale, propagating the fitted parameter covariance matrix

through the source covariance entries. They are asymptotic Wald intervals for estimation uncertainty under the declared random-effects model and allocation. Finite sampling of facet levels is part of that model; the intervals are not prediction intervals for the realized scores of a new panel and do not quantify misspecification of the facet populations. Wald theory does not generally apply to a source resting on a variance boundary. When inference uses an interior parameter block (a Gaussian fit whose joint curvature is indefinite at a boundary, or a discrete fit with any source on or near its boundary), the remaining intervals condition on the listed fixed components, and the fit says which way: a source whose fitted covariance is entirely zero is held at zero, and a singular but nonzero source is held fixed at its fitted covariance, which is a different statement. Unconditional coverage after that selection has not been established.

Discrete information is opt-in through `gt_control(discrete = list(check_hessian = TRUE))`. The covariance calculation inverts the negative-log-likelihood information matrix, including intercepts or thresholds, before propagating covariance-parameter uncertainty. It requires an accepted fit, stable finite differences and positive definite information. A source on or near its estimated covariance boundary is held fixed at its fitted covariance, as above; failed information checks leave intervals unavailable with a reason. Components explicitly supplied as fixed have no estimated uncertainty; any resulting intervals condition on those supplied values. Discrete intervals inherit first-order Laplace approximation error; general finite-sample calibration has not been established, and the coverage of discrete intervals conditional on a boundary source has not been assessed. Small-panel Wald intervals can be overly conservative. Coverage among available intervals must be reported separately from numerical acceptance and interval availability over all attempted panels. The lme4 Gaussian routes supply no parameter covariance matrix for these intervals.

With `bootstrap`, each coefficient is recomputed from every accepted replicate's source covariance matrices under the same counts, fixed facets and weights, and its standard error and interval come from those values; see `gt_bootstrap`. These intervals describe object sampling conditional on the observed instrumentation levels, not the sampling of the facet levels. They use neither the delta method nor the logit scale, and they do not condition on boundary sources, which every replicate re-estimates.

Conditioning on an estimated covariance boundary does not guarantee coverage at a true boundary. A sample can produce a positive interior estimate when the true variance is zero; its logit-Wald coefficient interval then excludes zero. The fitted-boundary rule does not identify every true boundary.

Coefficients for discrete outcomes are identified latent-response coefficients. They describe the average latent response under the declared measurement design, not binary proportions, majority-vote labels, or observed ordinal scores. Use `gt_categorical_target` for explicitly defined observed-score, category-vector or decision-agreement targets; its simulation errors are not parameter confidence intervals.

A nominal (unordered categorical) outcome with K categories has K - 1 latent contrast dimensions against its reference category. Its coefficients are the generalized eigenvalues of the universe-score covariance relative to the universe-plus-error covariance on those dimensions. `Erho2` and `Phi` report their mean. The range of the relative coefficient is in the columns `Erho2_min` and `Erho2_max`, the range of the absolute coefficient in the columns `Phi_min` and `Phi_max`, and the number of contrasts, K - 1, in the column `latent_dimensions`. The residual is the Gumbel noise of the K category utilities, $(\pi^2/6)(I + 11')$ on the contrasts. Generalized eigenvalues do not change under any invertible change of the contrast basis, so these coefficients do not depend on the reference category or on how categories are labelled; the largest is the reliability of the best linear composite of the contrasts (Joe and Woodward, 1976), and with two categories the coefficient is the binary logit one. This requires a covariance model that is itself unchanged by a change of reference: `covariance =`

"exchangeable" or "unstructured". A diagonal covariance on the contrasts is a different model for each reference category, so its coefficient is refused. Under the exchangeable structure every eigenvalue coincides. Interval bounds are given for the mean only, from the observed information by the delta method. A weighted composite including a nominal outcome is refused, since a nominal outcome has no numeric score; other outcomes in the same fit keep their own coefficients, and the range columns repeat a single-dimension outcome's coefficient. Ordinary coefficients for an incomplete allocation remain unavailable. An observed-row Gaussian fit can instead use gt_dstudy with projection = "balanced" to declare a complete future allocation; this is a separate target and requires every facet count. Instrumentation facets are averaged as random sources unless fixed names them; the function never infers a fixed-facet score universe on its own.

Value

A gt_reliability list containing per-outcome Erho2 and Phi with their standard errors and interval bounds, optional composite coefficients, for a fit with a nominal outcome the number of contrast dimensions and the smallest and largest generalized eigenvalue of each coefficient (see Details), universe, relative-error and absolute-error covariance matrices on the latent dimensions, the role each source played in the decomposition, coefficient scale, allocation, fixed facets, extrapolation flag, the uncertainty record, and fit diagnostics. With the default delta method, standard errors and bounds are NA whenever the fit carries no usable parameter covariance matrix; with bootstrap, they come from the accepted replicates and do not need that matrix. Printing shows a concise coefficient table, omits all-missing interval columns, and returns the complete object invisibly.

References

Brennan, R. L. (2001). *Generalizability Theory*. Springer. doi:10.1007/9781475734560

Joe, G. W., and Woodward, J. A. (1976). Some developments in multivariate generalizability. *Psychometrika*, 41, 205–217.

See Also

gt_score, gt_dstudy, gt_categorical_target, gaussian_observed

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
gt_reliability(fit)$per_trait
gt_reliability(fit, counts = c(rater = 6))
as.data.frame(gt_reliability(fit))
```

gt_report *Create and Export a Portable Aggregate Analysis Report*

Description

Collects retained analysis metadata and verified coefficient projections in a versioned snapshot, then exports a self-contained HTML report with aggregate tables, inline figures, limitations, and provenance. No model fitting, network access, external assets, or additional packages are needed.

Usage

```
gt_report(fit, preflight = NULL, reliability = NULL, dstudy = NULL,
  bootstrap = NULL)
gt_export_report(report, path, overwrite = FALSE)
## S3 method for class 'gt_report'
print(x, ...)
```

Arguments

fit	A gt_fit. Rejected fits can be reported as diagnostic objects, but their reliability or decision-study coefficients are refused.
preflight	Optional preflight report with matching family, source, grouping and counts. Outcome and category coding must also match.
reliability	Optional gt_reliability whose projections agree with the supplied fit under its declared allocation, scale, fixed facets, score, and uncertainty settings.
dstudy	Optional gt_dstudy checked against the supplied fit.
bootstrap	The gt_bootstrap result whose replicates gave reliability or dstudy their object-bootstrap intervals. It must have been made from fit; it is required when either table uses it and otherwise only checked against the fit.
report, x	A gt_report snapshot.
path	Destination .html or .htm file. The parent directory must already exist. The report embeds its styles, tables and SVG figures.
overwrite	Replace an existing destination only when explicitly TRUE. Defaults to FALSE.
...	Reserved for print-method compatibility.

Details

Association checks. Supplied coefficient objects are checked by recalculating their declared projections from the supplied fit. This reuses stored covariance estimates and uncertainty; it does not rerun an optimizer. Coefficient values, intervals, scales, allocations, fixed facets, composite weights and uncertainty context must agree. A table with object-bootstrap intervals is recalculated with the supplied bootstrap, which must belong to the fit; the snapshot keeps only its aggregate counts, never its replicate covariances, attempt records or seeds. Legitimately reordered result rows are accepted. This verifies numerical compatibility, not historical object identity or a shared original data file.

Preflight correspondence checks the available model specification, category coding, sources, free covariance-parameter counts when recorded, observed counts, replication and grouping scopes. Requested controls, fixed covariance values and resource-budget correspondence are not verified. Legacy preflight objects lacking grouping metadata are labelled as not verified for nesting. Even matching aggregate specifications cannot prove that preflight and fitting used the same observations; the report states that limitation prominently. No preflight is inferred from a fit and no coefficient section is calculated when its input is omitted.

Privacy scope. Only explicit aggregate fields are copied. Observations, facet-level identifiers, raw category labels, calls, models, grouping examples, arbitrary controls, diagnostic messages and filesystem paths are excluded. In particular, calls containing literal data frames and unused factor levels in empty example tables are not copied. Variable, outcome and source names, aggregate counts, and numerical estimates remain. These names and small aggregate counts can themselves be sensitive, so this is not anonymization; review a report before sharing it.

Categorical profiles use `category_1`, `category_2`, and so on in declared category order. The private label correspondence is not exported. Category counts and proportions of groups with each category are descriptive, not adequacy or identification thresholds. Gaussian minimum/maximum values, which are individual observed responses, are not copied.

Provenance. The fitting-session record contains only its retained R, platform and selected package versions. The report-generation environment is a separate record and never fills a missing fitting-session version. Session versions describe loaded packages, not necessarily the executing sources when code was sourced separately. Neither record is an archive hash, source commit, or release qualification. Missing session or retention records stay unknown. Model and data retention are not required when the fit retains the compact panel summary and source covariances needed for coefficient checks.

Figures and export. HTML text and SVG labels are escaped; figures use numeric coordinates and contain no scripts or remote references. Reliability figures show at most 30 coefficient rows; D-study figures show at most 600 rows. Coverage figures show at most 20 categories and 12 variables. Figure captions disclose these limits; complete aggregate tables remain available. Long graphical labels may be shortened, with complete labels in the tables. Source variance rows flag user-fixed covariance components so they cannot be mistaken for estimated zeros or estimated variance boundaries. Allocations are unconnected points; intervals appear only where recorded and available. Default discrete delta-method intervals require usable observed information and retain their first-order Laplace and boundary qualifications. Explicit bootstrap intervals retain their separate object-sampling, fixed-instrumentation and accepted-refit qualifications. The report cannot establish observed-label accuracy, a globally optimal allocation, statistical identification, or qualification of a private backend.

The exporter validates the supported schema version, prepares the complete HTML in a temporary file, and protects existing destinations unless `overwrite = TRUE`. No R objects or executable serialization are embedded in the HTML. Saving an RDS snapshot is a separate explicit operation.

Value

`gt_report()` returns a plain-list snapshot of class `gt_report`, with `schema_version = "1.0"`. It supports `saveRDS()` and `readRDS()` without retaining the fitted object. The main fields are:

- `generated_at`: report creation time.
- `analysis`: estimator, backend, numerical status and batch status.

- `design` and `families`: model declarations.
- `random_sources` and `nesting`: grouping declarations.
- `settings`: selected scalar controls.
- `covariance_settings`: recorded covariance structures.
- `diagnostics`: staged numerical status.
- `source_variances`: component values and fixed flags.
- `uncertainty`: interval scope and availability.
- `retention` and `provenance`: retained context.
- Optional `preflight`, `reliability` and `dstudy` tables.
- `bootstrap`: when a table uses object-bootstrap intervals, their methods, the resampled unit, the accepted, attempted and requested refit counts, and their scope; `NULL` otherwise.
- `association`, `privacy` and `limitations`: scope text.

Tables have atomic columns; absent optional sections are `NULL`. `gt_export_report()` returns the normalized output path invisibly. The print method returns the snapshot invisibly.

See Also

`gt_preflight`, `gt_reliability`, `gt_dstudy`, `gt_diagnostics`

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE)))
report <- gt_report(fit, gt_preflight(d, "score", design),
  gt_reliability(fit), gt_dstudy(fit, data.frame(rater = c(2, 3, 6))))
report
path <- tempfile(fileext = ".html")
gt_export_report(report, path)
unlink(path)
```

---

gt_score *Declare Composite-Score Weights*

Description

Stores explicit weights without normalizing or choosing a score scale.

Usage

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

Arguments

weights	Finite numeric vector with unique outcome names and at least one nonzero entry. Every modeled outcome must be weighted exactly once when used in a reliability calculation.
x	A gt_score object for the print method.
...	Further print arguments; currently unused.

Details

Weights must be substantively meaningful on the chosen outcome scale. A latent binary/ordinal composite is not an observed-score composite. Negative and non-unit-sum weights are allowed.

Value

A gt_score object containing weights and the "as_supplied" normalization policy. print() shows the weights and returns the object invisibly.

See Also

gt_reliability, gt_dstudy

Examples

```
gt_score(c(efficacy = 0.5, safety = 0.5))
```

---

gt_simulate *Simulate a New Gaussian or Categorical Annotation Panel*

Description

Generates a complete future coded panel under a fitted or specified parameter scenario. Gaussian random effects preserve covariance between outcomes or latent dimensions. Categorical observations use their fitted links, thresholds and nominal reference contrasts, retaining the declared category labels. Distinct item count and instrumentation allocation are separate inputs.

Usage

```
gt_simulate(fit, n_items, counts = NULL, seed, components = NULL,
  means = NULL, max_rows = 1000000L)
```

Arguments

fit	An explicitly numerically accepted gt_fit object for Gaussian outcomes or for binary, ordinal, and nominal categorical outcomes. Retain the covariance structures, family definitions and either modelled data or panel summary; categorical fits also need their latent-dimension mapping and any fitted thresholds. Gaussian and categorical outcomes cannot be mixed.
n_items	Number of new distinct objects, an integer of at least two. A categorical fit with modelled fixed calls retains its fitted item count and grouping; the coded objects receive fresh effects.
counts	Optional named positive integer instrumentation counts for a categorical future panel. Gaussian simulation still requires at least two levels per facet. Unspecified counts retain the fitted panel counts. Counts for a nested child describe its local coded levels within each parent combination.
seed	Required integer seed between zero and .Machine$integer.max. The caller's random-number generator kind and state are restored, including when no random seed existed before the call.
components	Optional complete named list of covariance matrices, one for every retained source. Gaussian models include Residual; categorical models have no Gaussian residual component. Each matrix must be symmetric, positive semidefinite, and preserve its fitted structure. Gaussian matrix axes identify outcomes; categorical axes identify the fitted latent dimensions, including one contrast per nonreference nominal category. Fully unnamed axes follow fitted order; labelled axes must name every axis exactly once. The fitted components are used when NULL.
means	Optional finite named numeric vector. Gaussian entries name every outcome; categorical entries name every fitted latent dimension. Defaults to the fitted means or intercepts. Ordinal intercepts must remain zero because their location is represented by the retained fitted thresholds.
max_rows	Maximum generated rows, between one and one million. A separate estimated 512 MiB workspace limit also applies. Neither limit changes fitting controls or permits an unsupported model.

Details

Gaussian simulation requires a complete balanced fitted panel with one outcome vector per cell and no batch declaration. Categorical simulation supports univariate and joint binary, ordinal and nominal outcomes, explicit random sources, parent-scoped grouping terms and the declared equal cell replication. Missing whole cells in an accepted categorical fit do not prevent generating a complete future allocation; no original missing values are imputed. No globally unique physical child labels are generated. One future level of a categorical instrumentation facet is allowed, although estimating that facet's variance in a pilot requires at least two observed levels. gt_pilot_plan() therefore requires at least two levels in its pilot grid while allowing one level in the separately specified target protocol.

For categorical models, source effects are Gaussian on the fitted latent predictor scale. Binary probabilities use the fitted probit or logit link; ordinal probabilities are adjacent cumulative-probability

differences at the fitted thresholds; nominal probabilities use reference-category softmax. Observations and outcomes are independent conditional on those predictors; between-outcome dependence arises from the source covariance matrices. Nominal labels receive no arbitrary numeric scores. A mixed categorical model can have different links and numbers of categories across its outcomes.

Modelled equal fixed calls are supported for categorical fits with one row per cell. Each call effect is shared by its retained group of items under one complete facet condition. Instrumentation counts can change, but item count, group membership and batch size remain fixed. Recorded call and order columns are generated when declared. Unmodelled batch declarations, positional effects, variable calls and regrouping are refused.

Zero and singular positive semidefinite components are allowed. Only negative eigenvalues within a relative numerical tolerance of 10^{-10} (using a unit minimum scale) are set to zero. No positive interior jitter is added. A degenerate scenario can generate data that fitting subsequently refuses.

Every facet is sampled afresh; the function does not reuse estimated random effects or condition on observed evaluator or item realizations. This is design simulation. A fitted-model parameter scenario can also provide data for an explicitly designed parametric-bootstrap analysis, but this function performs no refitting, interval construction, boundary calibration, or coverage assessment.

Value

A data frame with integer coded object and facet columns. Gaussian outcomes are numeric; binary and nominal outcomes are factors, and ordinal outcomes are ordered factors. All declared category levels are retained even if a simulated sample contains no observations in one of them. Its `simulation` attribute records the seed, counts, means, covariance matrices, covariance structures, parameter source, and interpretation. Categorical results also record families, thresholds, latent dimensions, replication and any fixed-call layout. The generated covariance matrices, including numerical tolerance adjustments, are recorded explicitly.

See Also

`gt_pilot_plan`, `gt_fit`, `gt_dstudy`

Examples

```
set.seed(91)
panel <- expand.grid(item = 1:20, evaluator = 1:3)
panel$score <- rnorm(20)[panel$item] + rnorm(3)[panel$evaluator] +
  rnorm(nrow(panel))
fit <- gt_fit(panel, "score", gt_design("item", "evaluator"))
if (isTRUE(gt_diagnostics(fit)$numerically_accepted)) {
  simulated <- gt_simulate(fit, n_items = 30,
    counts = c(evaluator = 4), seed = 2026)
  head(simulated)
}
```

`plot.gt_reliability` *Plot Reliability Coefficients and Available Intervals*

Description

Draws a forest plot of the already calculated relative and absolute coefficients, with available estimation intervals and explicit outcome labels.

Usage

```
## S3 method for class 'gt_reliability'
plot(x, coefficient = c("Erho2", "Phi"),
  interval = TRUE, target = NULL, ...)
```

Arguments

`x`	A result from `gt_reliability()`.
`coefficient`	One or both of `"Erho2"` and `"Phi"`, without duplicates.
`interval`	Whether to draw available interval bounds. Missing bounds are never replaced by a zero-width interval.
`target`	Optional reference line at one number between zero and one. It is a descriptive target, not a confidence statement.
`...`	Named graphical arguments passed to `graphics::plot()`, such as `col`, `pch`, `main`, `xlim`, or `sub`, replacing defaults. The coefficient coordinates are supplied by the method.

Details

Outcome rows and a weighted composite have distinct labels. The horizontal axis names the observed or latent coefficient scale. The default subtitle identifies unavailable or hidden intervals, conditional uncertainty, and extrapolation. A caller overriding the subtitle should preserve the interpretation relevant to their analysis. For a design that declared batches with `gt_batch`, the plot also states above the plotting region that they are not modelled; that statement is not a subtitle and a caller's `sub` does not replace it. Unavailable estimates keep their label but have no plotted point. Changing the graphic does not refit the model or alter any acceptance or inference decision.

On narrow devices, long labels are shortened with an ellipsis and a bracketed row number referring to `as.data.frame(x)`. Full outcome names remain in that table. Margins are restored after plotting.

Default discrete delta-method intervals require optional full observed information that was requested and passed its acceptance, step-stability and curvature checks; at a boundary source they condition on that source held fixed. These asymptotic intervals inherit the first-order Laplace approximation. Finite-sample coverage is not established. An explicit `gt_bootstrap` can instead supply percentile or BCa intervals without observed information; these describe object sampling conditional on the observed instrumentation levels and on accepted refits, with their separate availability checks. Latent-response coefficients do not describe observed label proportions or majority votes. Interval and boundary qualifications are described in `gt_reliability`. The plotting method uses base graphics; PDF and PNG devices can save the same figure without additional plotting dependencies.

Value

Returns x invisibly.

See Also

gt_reliability, gt_dstudy, gt_preflight

Examples

```
# Entirely synthetic continuous scores.
d <- expand.grid(item = 1:12, evaluator = 1:3)
d$quality <- sin(d$item) + d$evaluator / 5 +
  cos(d$item * d$evaluator) / 4
fit <- gt_fit(quality ~ item + evaluator, d, measured = "item",
  control = gt_control(gaussian = list(retry_seed = 42)))
reliability <- gt_reliability(fit)
plot(reliability, target = 0.80)
as.data.frame(reliability)
```

---

print.summary.gt_fit *Print a Concise Generalizability Model Summary*

Description

Displays model scope, numerical acceptance, likelihood, source variances, and fixed estimates without printing the fitted backend model or observation data.

Usage

```
## S3 method for class 'summary.gt_fit'
print(x, ..., digits = max(3L,
  getOption("digits") - 3L))
```

Arguments

x	A summary.gt_fit object returned by summary(fit).
...	Additional arguments; currently unused.
digits	Number of significant digits for printed estimates, from 1 to 22.

Details

The printed summary reports the selected attempt, recorded failed attempts, acceptance failures, natural covariance boundaries, artificial parameter bounds, and likelihood-approximation status separately. If the native Gaussian transcript does not identify a unique selected trial, the output says "not identified". An accepted fit can have earlier failed attempts or a natural covariance boundary. A rejected fit's estimates are printed for diagnosis only; it remains ineligible for reliability or decision-study calculations.

The printed variance table shows at most 20 rows and the attempt-failure table at most 6 rows. Complete tables and covariance matrices remain in the summary.

Use `gt_diagnostics(fit)` for full diagnostic details.

Fixed location or contrast estimates are printed on the fitted family's scale. Numerical acceptance and first-order Laplace approximation adequacy remain separate; the summary makes no additional claim of statistical validation.

Value

Returns `x` invisibly. The summary object remains a list with outcome and family specifications, source covariance matrices, a source-variance table, means or coefficients, thresholds, likelihood, and diagnostics.

See Also

`gt_fit`, `gt_diagnostics`, `gt_components`

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
s <- summary(fit)
print(s, digits = 3)
s$variances
```

Index

∗ **datasets**
gtheory_datasets, 7
gtheory_drug_review, 10
gtheory_hate_speech, 11
gtheory_mental_health, 13

as.data.frame, *65*
as.data.frame.gt_batch_dstudy *(*gt_batch_dstudy*)*, 18
as.data.frame.gt_batch_reliability *(*gt_batch_reliability*)*, 19
as.data.frame.gt_bootstrap *(*gt_bootstrap*)*, 22
as.data.frame.gt_categorical_target *(*gt_categorical_target*)*, 25
as.data.frame.gt_dstudy *(*gt_dstudy*)*, 45
as.data.frame.gt_pilot_plan *(*gt_pilot_plan*)*, 63
as.data.frame.gt_plan *(*gt_plan*)*, 68
as.data.frame.gt_reliability *(*gt_reliability*)*, 80

barplot, *74*

coef.gt_fit *(*gt_fit_methods*)*, 56

Dataset overview, *11*, *13*, *15*, *50*
dataset overview, *4*, *10–14*, *49*
deviance.gt_fit *(*gt_fit_methods*)*, 56
Drug-Review, *4*, *49*
Drug-Review data, *9*, *50*
Drug-Review data topic, *8*
drug_review *(*gtheory_drug_review*)*, 10
drug_review_4aspect *(*gtheory_drug_review*)*, 10

formula.gt_fit *(*gt_fit_methods*)*, 56

gaussian_observed, 5, *31*, *40*, *53*, *84*
glm, *5*
gt_batch, 15, *21*, *39*, *40*, *47*, *53*, *76*, *82*, *91*
gt_batch_dstudy, *17*, 18, *21*, *82*
gt_batch_reliability, *4*, *17–19*, 19, *82*
gt_bootstrap, *4*, 22, *31*, *45*, *47*, *55*, *81*, *83*, *85*, *91*
gt_categorical_target, *4*, 25, *63–66*, *68–70*, *72*, *82–84*
gt_component_vcov *(*gt_fit_methods*)*, 56
gt_components, 29, *35*, *59*, *93*
gt_control, *6*, *23*, 30, *42*, *44*, *53–55*, *64*, *65*, *74*, *78*
gt_design, *5*, *16*, *17*, 38, *47*, *55*, *73*, *82*
gt_diagnostics, *29*, *38*, 41, *55*, *59*, *62*, *79*, *87*, *93*
gt_dstudy, *5*, *6*, *17*, *19*, *22*, *25*, 45, *68*, *72*, *79*, *84*, *87*, *88*, *90*, *92*
gt_dstudy_target, *72*
gt_dstudy_target *(*gt_dstudy*)*, 45
gt_example, *5*, *7*, *9–11*, *13*, *15*, 48
gt_export_report *(*gt_report*)*, 85
gt_family, *11*, *13*, *15*, *40*, *50*, 50, *73*
gt_fit, *5*, *6*, *9*, *25*, *29*, *32*, *38*, *40*, *44*, *50*, *51*, 51, *57*, *59*, *62*, *73*, *74*, *79*, *90*, *93*
gt_fit_methods, *53*, 56
gt_laplace_check, *53*, 60
gt_pilot_plan, *4*, 63, *90*
gt_plan, *4*, *28*, 68
gt_preflight, *5*, *17*, *40*, 73, *87*, *92*
gt_reliability, *4*, *9*, *17*, *21*, *22*, *25*, *28*, *45*, *46*, *48*, *58*, *63*, *64*, *68*, *70*, *72*, *78*, *79*, 80, *87*, *88*, *91*, *92*
gt_report, *17*, 85
gt_score, *15*, *21*, *48*, *64*, *84*, 87
gt_simulate, *4*, *28*, *63*, *64*, *68*, 88
Gtheory4LLM *(*Gtheory4LLM-package*)*, 3
Gtheory4LLM-datasets *(*gtheory_datasets*)*, 7
Gtheory4LLM-package, 3
gtheory_datasets, 7
gtheory_drug_review, 10

gtheory_hate_speech, 11
gtheory_mental_health, 13

Hate-Speech, *4*, *49*
Hate-Speech data, *9*, *50*
Hate-Speech data topic, *8*
hate_speech *(*gtheory_hate_speech*)*, 11

image, *74*

logLik.gt_fit *(*gt_fit_methods*)*, 56

Mental-Health, *4*, *49*
Mental-Health data, *9*, *50*
Mental-Health data topic, *8*
mental_health_3group
*(*gtheory_mental_health*)*, 13
mental_health_3L
*(*gtheory_mental_health*)*, 13
mental_health_6flag
*(*gtheory_mental_health*)*, 13
mental_health_7L
*(*gtheory_mental_health*)*, 13
mental_health_nominal
*(*gtheory_mental_health*)*, 13

nobs.gt_fit *(*gt_fit_methods*)*, 56

plot.gt_dstudy *(*gt_dstudy*)*, 45
plot.gt_preflight *(*gt_preflight*)*, 73
plot.gt_reliability, 91
print method, *55*
print.gt_batch *(*gt_batch*)*, 15
print.gt_batch_dstudy
*(*gt_batch_dstudy*)*, 18
print.gt_batch_reliability
*(*gt_batch_reliability*)*, 19
print.gt_bootstrap *(*gt_bootstrap*)*, 22
print.gt_categorical_target
*(*gt_categorical_target*)*, 25
print.gt_control *(*gt_control*)*, 30
print.gt_design *(*gt_design*)*, 38
print.gt_diagnostic_stage
*(*gt_diagnostics*)*, 41
print.gt_diagnostic_stages
*(*gt_diagnostics*)*, 41
print.gt_diagnostics *(*gt_diagnostics*)*,
41
print.gt_dstudy *(*gt_dstudy*)*, 45
print.gt_example *(*gt_example*)*, 48
print.gt_family *(*gt_family*)*, 50
print.gt_fit *(*gt_fit*)*, 51
print.gt_laplace_check
*(*gt_laplace_check*)*, 60
print.gt_pilot_plan *(*gt_pilot_plan*)*, 63
print.gt_plan *(*gt_plan*)*, 68
print.gt_preflight *(*gt_preflight*)*, 73
print.gt_reliability *(*gt_reliability*)*,
80
print.gt_report *(*gt_report*)*, 85
print.gt_score *(*gt_score*)*, 87
print.summary.gt_fit, 92

quantile, *24*

summary.gt_bootstrap *(*gt_bootstrap*)*, 22
summary.gt_fit *(*gt_fit*)*, 51

update.formula, *57*
update.gt_fit *(*gt_fit_methods*)*, 56

vcov.gt_fit *(*gt_fit_methods*)*, 56