

Package ‘Gtheory4LLM’

October 4, 2026

Type Package

Title Generalizability Theory for LLM Subjective Tasks

Version 0.2.0

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; discrete models use a dense 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). Gaussian fits report asymptotic Wald standard errors for their variance components and delta-method intervals for the coefficients; discrete fits report point estimates only. Scalar nominal reliability and joint Gaussian-discrete fitting are not implemented. Includes three publicly archived LLM annotation datasets covering hate-speech, mental-health, and drug-review tasks. Discrete fitting is limited to small models; 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](https://doi.org/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, stats, utils

Suggests lme4, ordinal, knitr, rmarkdown

VignetteBuilder knitr

URL <https://github.com/Veronica0206/Gtheory4LLM>
BugReports <https://github.com/Veronica0206/Gtheory4LLM/issues>
Collate 'design.R' 'family.R' 'gaussian_retry.R' 'gaussian_engine.R'
 'gaussian.R' 'discrete_response.R' 'discrete_dense.R'
 'discrete_sparse.R' 'discrete_mode.R' 'discrete_sparse_mode.R'
 'discrete.R' 'preflight.R' 'diagnostics_stages.R' 'fit.R'
 'methods.R' 'reliability.R' 'examples.R'
NeedsCompilation no
Repository CRAN
Date/Publication 2026-10-04 09:30:02 UTC

Contents

Gtheory4LLM-package	2
gtheory_datasets	4
gtheory_drug_review	7
gtheory_hate_speech	9
gtheory_mental_health	10
gt_components	13
gt_control	14
gt_design	18
gt_diagnostics	20
gt_dstudy	22
gt_example	24
gt_family	26
gt_fit	27
gt_fit_methods	29
gt_preflight	31
gt_reliability	33
gt_score	35
print.summary.gt_fit	36
Index	38

Gtheory4LLM-package	<i>Generalizability Theory for LLM Subjective Tasks</i>
---------------------	---

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. Binary, ordinal, and unordered categorical outcomes use their specified observation distributions with Gaussian latent random effects, fitted by a dense first-order Laplace approximation. Joint Gaussian models and joint discrete models are supported; a joint Gaussian-discrete likelihood is not implemented.

Observed Gaussian reliability and latent binary or ordinal reliability are distinct estimands. Scalar nominal reliability is not implemented. Decision studies require a complete balanced coded panel. Their projections depend on fitted source covariances and the specified universes of evaluators and prompts. Adding arbitrarily chosen LLMs does not ensure those assumptions hold. Gaussian projections carry delta-method intervals for estimation uncertainty in the fitted source covariances; they do not quantify sampling uncertainty in the facet populations, automatically find a minimum allocation, or establish annotation accuracy against human judgments. Discrete projections are point estimates only.

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, pre-processing, 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](https://doi.org/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.

Gaussian fits report asymptotic Wald standard errors for their source variance components and delta-method intervals for G and Phi; see [gt_reliability](#). Bootstrap and jackknife intervals are not implemented, and the discrete Laplace engine reports point estimates only.

Why this package declares R ($\geq 4.5.0$): Nothing in this package's own R code needs R 4.5. The floor comes from the OpenMx version it supports. OpenMx 2.22.11 calls `Rf_isDataFrame`, a C entry point R introduced in 4.5.0, while declaring only R ($\geq 3.5.0$) for itself. Declaring the floor here is what prevents an install on an older R from failing later with an unexplained compilation error. Earlier combinations such as R 4.3 with OpenMx 2.21.11 have been reported to run this package's checks; that pairing is not part of its validated gate. To use one, install from source with a relaxed floor, or load the implementation directly by sourcing 'load_functions.R', which imposes no version requirement of its own.

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](#)

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 = "manuscript", 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](https://doi.org/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](https://doi.org/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 = "manuscript" 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](https://doi.org/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](https://doi.org/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](https://doi.org/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 = "manuscript", 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 numeric manuscript 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](https://doi.org/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](https://doi.org/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](https://doi.org/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 = "manuscript")
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 = "manuscript" 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](https://doi.org/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](https://doi.org/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_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 <code>gt_fit</code> 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())
```

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.

Details

Retained components. retain decides which optional parts of a fitted object are kept. All four default to TRUE, so a fit keeps exactly what it kept before these switches existed; dropping a component is always something the caller asked for.

- **data:** the modelled data frame. 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.
- **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:

- 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).
- max_preparation_bytes: 512 * 1024^2.

The allocation guard estimates preparation memory, excluding caller data and downstream OpenMx objects. An explicit retry seed preserves the caller's random-number state. 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:

- max_random_dimension: 200.
- max_observations: 1200.
- max_parameters: 80.
- max_dense_bytes: 512 * 1024^2.

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 "nllminb".
- start: NULL (automatic response and covariance starts).
- 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 "log_cholesky", retaining all cross-outcome covariance parameters. Explicit "variance" and "log_cholesky" selections remain available. Direct variance coordinates admit exact zeros; an explicit "variance" request with joint unstructured covariance is rejected rather than removing covariances. The variance upper bound is $\exp(10)$, matching the largest diagonal variance allowed

by the log-Cholesky bounds. `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$parameters` vector can initialize a refit of the same model, with unchanged family links, category coding, covariance structure, and covariance parameterization. These are optimizer coordinates: ordinal threshold parameters contain the first threshold followed by log positive gaps; log-Cholesky diagonal parameters are log standard deviations; direct-variance parameters are variances. Actual and automatic initial vectors are retained in the diagnostics, and `fit$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.

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 $\max(200, 10 * \text{maxit})$, 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, not an artificial optimizer floor. Within `gt_diagnostics(fit)$diagnostics`, the `zero_variance_parameters` field lists exact zeros. 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 control selection; `covariance_parameterization` records the resolved coordinates ("variance", "log_cholesky", 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.

Setting alternative starts to zero disables numerical acceptance, not just an optional report. The discrete optimization-attempt budget is $2 * (1 + \text{alternative_starts})$: 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.

`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.

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.

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 = "nllminb", 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)
```

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. 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.

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`.

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. Gaussian fitting, `gt_reliability()` and `gt_dstudy()` all 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.

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 this backend rejects that panel instead of guessing which cells were never possible. 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 meta-data, and pending data-validation information.

See Also

[gt_fit](#), [gt_family](#)

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$terms_requested
# The same crossed design without the observation-level source, which a
# binary or ordinal fit at one observation per cell cannot identify. This is
# where a first discrete fit should start; see the worked script in
# system.file("examples", "discrete-first.R", package = "Gtheory4LLM").
discrete_ready <- gt_design("item", "rater", full_cell = FALSE)
discrete_ready$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

```

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.

Passing these checks does not prove structural identification, a global optimum, or adequate approximation in sparse or strongly dependent data.

`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 for a Gaussian fit whose numerically differentiated Hessian was computed and is positive definite. The 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 discrete Laplace engine computes no observed information, so its standard errors are always unavailable and its coefficients remain point estimates. 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.

`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, and data-validation notes.

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:

- `optimizer`: the selected optimization engine.
- `optimization_trials`: the recorded trial count.
- `selected_attempt`: the returned candidate's identifier.
- `acceptance_failures`: reasons for rejecting the returned fit.
- `attempt_failures`: recorded trial errors or rejections.

- `boundary_sources`: zero or nearly singular covariance sources.
- `parameter_bounds`: artificial parameter-bound contacts.
- `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)
## S3 method for class 'gt_dstudy'
print(x, ..., digits = 4L, max_rows = 12L)
## S3 method for class 'gt_dstudy'
plot(x, coefficient = "Erho2", interval = TRUE, ...)
```

Arguments

<code>fit</code>	A numerically accepted <code>gt_fit</code> with a complete balanced coded panel.
<code>grid</code>	A nonempty data frame of positive integer facet counts. Columns name declared facets; omitted facets retain their fitted counts.
<code>scale</code>	"observed" for Gaussian outcomes or explicitly "latent" for binary/ordinal outcomes.
<code>score</code>	Optional composite weights created by <code>gt_score()</code> .

<code>fixed</code>	Instrumentation facets treated as fixed, as in <code>gt_reliability()</code> . A fixed facet may not appear in <code>grid</code> : its universe is exactly its observed levels.
<code>level</code>	Confidence level for the reported coefficient intervals.
<code>x</code>	A <code>gt_dstudy</code> object.
<code>coefficient</code>	Plot "Erho2" (default) or "Phi".
<code>interval</code>	Draw the interval bounds as vertical segments when the fit supplies them.
<code>digits</code>	Significant digits used when printing.
<code>max_rows</code>	Maximum number of coefficient rows to print.
<code>...</code>	Additional arguments passed to <code>graphics::plot()</code> , or reserved for printing.

Details

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. Reported 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. Intervals are unavailable for discrete fits. 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)))
study$results
plot(study)
```

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", "manuscript"),
  directory = NULL)
```

Arguments

name	Outcome-set name, or NULL to list available sets.
coding	Use "native" to preserve response types. Use "manuscript" for the seven original Gaussian working-score codings.
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.

Details

Every set retains 21,600 measurement rows: 100 text objects by 4 evaluators, 3 prompts, 6 temperatures, and 3 seeds. Full tables exceed default dense discrete fitting limits; use a predeclared small panel or a future scalable backend for 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 manuscript-score 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 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.

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	<i>Specify a Gaussian, Binary, Ordinal, or Unordered Categorical Outcome</i>
-----------	--

Description

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

Usage

```
gt_family(family = "gaussian", link = NULL, levels = NULL,
          reference = NULL)
```

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.

Details

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.

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")
```

Description

Fits selected random-source covariance models using an exact balanced Gaussian likelihood or a dense first-order Laplace likelihood for discrete outcomes.

Usage

```
gt_fit(data, outcomes, design, family = gt_family("gaussian"),
       estimator = NULL, covariance = "unstructured", residual = NULL,
       control = gt_control())
## 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.
outcomes	One or more unique outcome column names.
design	A declaration created by <code>gt_design()</code> .
family	One <code>gt_family</code> 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.
residual	The Gaussian residual structure. Choose "unstructured", "diagonal", or "pooled"; pooled gives equal outcome variances. Omit for discrete outcomes.
control	A <code>gt_control</code> object.
x, object	A fitted <code>gt_fit</code> object for the print or summary method.
...	Further method arguments; currently unused.

Details

The Gaussian backend requires a complete coded Cartesian panel with one observation per cell and at most 4096 factorial strata. Implicit normalized Helmert contrasts avoid quadratic allocation along a large object axis. Gaussian ML profiles one fixed intercept per outcome; REML includes the unscaled-intercept constant $D * \log(N)$. 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. `legacy_AIC` and `legacy_BIC` preserve the archive's variance-only parameter convention. 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.

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.

Gaussian fitting also computes asymptotic Wald standard errors for the source variance components from the numerically differentiated restricted- or profile-likelihood Hessian. 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.

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.

Value

A `gt_fit` object containing source covariance matrices, estimates, likelihood and parameter counts, the resolved design and observation families, modeled data, and numerical diagnostics. Gaussian objects also retain the OpenMx model, prepared sufficient statistics, an uncertainty record holding the parameter covariance matrix and its delta-method mapping to source covariance entries, and a `component_standard_errors` table. 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, 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`; the example below shows how to source it.

Examples

```
d <- expand.grid(item = 1:12, rater = 1:3)
```

```

d$score <- sin(d$item) + d$rater / 5 + cos(d$item * d$rater) / 4
design <- gt_design("item", "rater", random = ~ item + rater)
fit <- gt_fit(d, "score", design,
  control = gt_control(gaussian = list(check_hessian = FALSE, retry_seed = 42)))
print(fit)
summary(fit)$minus2loglik
# 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
# 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, observation count, 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'
nobs(object, ...)
## 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	A fitted gt_fit object.
fit	A fitted gt_fit object.
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.
...	Further method arguments; currently 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. 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.

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. This implementation does not estimate it: Gaussian outcome means are profiled out of the likelihood rather than fitted as free parameters, so no curvature for them exists, and the dense discrete engine computes no observed information for anything. 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()` is the covariance that does exist: the sampling covariance of the estimated source covariances, which is what [gt_reliability](#) propagates into a coefficient interval. It raises an error carrying the fit's own recorded reason when none is available, rather than returning zeros or an invented matrix: a discrete fit always, a Gaussian fit computed with `check_hessian = FALSE`, and one whose Hessian is not positive definite. Where standard errors condition on boundary components being held fixed, `conditional_on_fixed` names them all and `conditional_on_zero` names the subset whose fitted covariance is entirely zero; a singular but nonzero source is held fixed at its fitted covariance, not at zero.

`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

`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. `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)
nobs(fit)
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_preflight

Inspect Data, Source Dimensions, and Backend Limits Before Fitting

Description

Reports observed design counts, random-source dimensions, parameter counts, and the current backend's structural and resource restrictions without optimizing a model. Uses the same category and source-resolution rules as fitting.

Usage

```
gt_preflight(data, outcomes, design, family = gt_family("gaussian"),
  covariance = "unstructured", residual = NULL,
  control = gt_control())

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

Arguments

data	A nonempty data frame with unique column names.
outcomes	One or more outcome column names.
design	A design from gt_design .
family	A family from gt_family , or a named family list matching the outcomes.

covariance	The source covariance specification accepted by <code>gt_fit</code> .
residual	Gaussian residual covariance specification, or NULL for the default. Do not specify this argument for discrete outcomes.
control	Engine controls from <code>gt_control</code> . Resource limits are read from the selected engine's controls.
x	An object returned by <code>gt_preflight</code> .
...	Reserved for print-method compatibility.

Details

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.

Gaussian checks require a complete coded Cartesian panel with one measurement per cell and resolve a full-cell source into the residual when appropriate. Discrete fitting can admit missing whole cells, but analytic reliability and D studies still require a complete balanced coded panel. Within configured size limits, preflight also applies the current discrete source-kernel checks. When an earlier structural or resource check fails, `kernel_check` explicitly reports that the kernel checks were skipped. Source counts then refer to the requested model if family-specific resolution failed.

The Gaussian allocation estimate covers preparation arrays only. The discrete byte count covers one dense random-design matrix and one random-effect Hessian only. Neither predicts peak memory or runtime. Fewer than five observed groups triggers a descriptive note, not a rejection rule. Starting values and optimizer-specific controls are validated by fitting.

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 responses. Latent coefficients describe averages of latent responses, not majority-vote decisions or observed label proportions. No scalar nominal coefficient is implemented.

Value

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

- `sources`: a data frame describing the random sources.
- `checks`: a data frame of the structural and resource checks.
- `fitting_feasible`: whether those checks passed.
- `supported_reliability_scales`: conditional coefficient scales.

Feasibility is not a fitting or identification result. Invalid input types or observation categories raise an error. Unsupported observed designs and exceeded resource limits are reported as failed checks.

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 Gaussian fitting and analytic reliability.
incomplete <- gt_preflight(d[-1, ], "score", design)
incomplete$checks[!incomplete$checks$passed, ]
```

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

Arguments

fit	A numerically accepted <code>gt_fit</code> with a complete balanced coded panel.
scale	"observed" for Gaussian outcomes (default); explicitly request "latent" for binary/ordinal outcomes.
score	Optional <code>gt_score</code> 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.

x	A gt_reliability object.
digits	Significant digits used when printing.
max_rows	Maximum number of outcome rows to print.
...	Reserved for additional print arguments.

Details

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, 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. The discrete engine supplies no parameter covariance matrix, so its coefficients remain point estimates.

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. Scalar reliability for unordered categories, observed discrete reliability, and unbalanced decision-study integration are not implemented. 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, universe/relative-error/absolute-error covariance matrices, the role each source played in the decomposition, coefficient scale, allocation, fixed facets, extrapolation flag, the uncertainty record, and fit diagnostics. Standard errors and bounds are NA whenever the fit carries no usable parameter covariance 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](https://doi.org/10.1007/9781475734560)

See Also

[gt_score](#), [gt_dstudy](#)

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))
```

gt_score

Declare Composite-Score Weights

Description

Stores explicit weights without normalizing or choosing a score scale.

Usage

```
gt_score(weights)
```

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.
---------	---

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.

See Also

[gt_reliability](#), [gt_dstudy](#)

Examples

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

```
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, 4
 - gtheory_drug_review, 7
 - gtheory_hate_speech, 9
 - gtheory_mental_health, 10
- coef.gt_fit(gt_fit_methods), 29
- Dataset overview, 8, 10, 12, 25
- dataset overview, 3, 7–12, 25
- Drug-Review, 3, 25
- Drug-Review data, 7, 25
- Drug-Review data topic, 6
- drug_review(gtheory_drug_review), 7
- drug_review_4aspect
(gtheory_drug_review), 7
- glm, 4
- gt_component_vcov(gt_fit_methods), 29
- gt_components, 13, 30, 31, 37
- gt_control, 14, 22, 28, 32
- gt_design, 4, 18, 23, 28, 31, 34
- gt_diagnostics, 13, 17, 20, 28, 31, 33, 37
- gt_dstudy, 4, 22, 33, 35, 36
- gt_example, 4, 7–10, 12, 24
- gt_family, 8, 10, 12, 19, 25, 26, 31
- gt_fit, 4, 6, 13, 17, 19, 22, 25, 26, 27, 30–33, 37
- gt_fit_methods, 29
- gt_preflight, 4, 31
- gt_reliability, 3, 6, 23, 30, 33, 33, 36
- gt_score, 12, 23, 35, 35
- Gtheory4LLM(Gtheory4LLM-package), 2
- Gtheory4LLM-datasets
(gtheory_datasets), 4
- Gtheory4LLM-package, 2
- gtheory_datasets, 4
- gtheory_drug_review, 7
- gtheory_hate_speech, 9
- gtheory_mental_health, 10
- Hate-Speech, 3, 25
- Hate-Speech data, 7, 25
- Hate-Speech data topic, 6
- hate_speech(gtheory_hate_speech), 9
- logLik.gt_fit(gt_fit_methods), 29
- Mental-Health, 3, 25
- Mental-Health data, 7, 25
- Mental-Health data topic, 6
- mental_health_3group
(gtheory_mental_health), 10
- mental_health_3L
(gtheory_mental_health), 10
- mental_health_6flag
(gtheory_mental_health), 10
- mental_health_7L
(gtheory_mental_health), 10
- mental_health_nominal
(gtheory_mental_health), 10
- nobs.gt_fit(gt_fit_methods), 29
- plot.gt_dstudy(gt_dstudy), 22
- print method, 28
- print.gt_diagnostic_stage
(gt_diagnostics), 20
- print.gt_diagnostic_stages
(gt_diagnostics), 20
- print.gt_diagnostics(gt_diagnostics), 20
- print.gt_dstudy(gt_dstudy), 22
- print.gt_fit(gt_fit), 27
- print.gt_preflight(gt_preflight), 31
- print.gt_reliability(gt_reliability), 33
- print.summary.gt_fit, 36
- summary.gt_fit(gt_fit), 27
- vcov.gt_fit(gt_fit_methods), 29