Package 'aiEvalR'

August 30, 2026

Type Package

Title Statistical and Psychometric Evaluation of AI Systems

Version 0.1.0

Description
Evaluates artificial intelligence (AI) systems as measurement instruments using psychometric methods. Provides multi-facet generalizability theory (G-study and D-study) via 'lme4', reliability via the intraclass correlation coefficient (ICC), calibration via the expected calibration error (ECE) and Brier score, robustness stress testing, and group disparity diagnostics. Item-level differential item functioning (DIF) based on item response theory (IRT) is delegated to the 'aiDIF' package. Methods follow Cronbach, Gleser, Nanda and Rajaratnam (1972, <ISBN:9780471188506>) and Brennan (2001) <doi:10.1007/978-1-4757-3456-0>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports stats, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown, covr, boot, lme4, dplyr, ggplot2, aiDIF, spelling

Config/testthat/edition 3

URL https://github.com/causalfragility-lab/aiEvalR

BugReports https://github.com/causalfragility-lab/aiEvalR/issues

VignetteBuilder knitr

NeedsCompilation no

Author Subir Hait [aut, cre] (ORCID: <https://orcid.org/0009-0004-9871-9677>)

Maintainer Subir Hait <haitsubi@msu.edu>

Config/roxygen2/version 8.0.0

Repository CRAN

Date/Publication 2026-08-30 09:10:27 UTC

Contents

`adversarial_stability` *Adversarial stability*

Description

Compares AI outputs on original vs. adversarially-perturbed inputs (e.g., injected distractors, minor semantic-preserving edits designed to probe brittleness) and reports the proportion of cases where the AI's substantive conclusion flips.

Usage

```
adversarial_stability(original, adversarial, flip_fn = function(a, b) a != b)
```

Arguments

`original`	Vector of original outputs (e.g., discrete labels or scores).
`adversarial`	Vector of outputs on adversarial inputs, aligned with `original`.
`flip_fn`	Function taking `(original_i, adversarial_i)` and returning `TRUE` if the conclusion flipped. Default: exact inequality (appropriate for discrete labels).

Value

A list with `flip_rate` and `n`.

---

ai_bootstrap_reliability *Bootstrap confidence interval for AI reliability estimates*

Description

Nonparametric bootstrap (resampling prompts with replacement) for the ICC computed by `ai_test_retest()`, giving a confidence interval rather than a point estimate alone.

Usage

```
ai_bootstrap_reliability(responses, n_boot = 1000, conf = 0.95, seed = NULL)
```

Arguments

`responses`	Numeric matrix: rows = prompts, columns = occasions.
`n_boot`	Integer. Number of bootstrap resamples. Default 1000.
`conf`	Numeric. Confidence level. Default 0.95.
`seed`	Optional integer seed for reproducibility. Default `NULL` (no seed set; caller's RNG state is used as-is).

Value

A list with `estimate`, `ci_lower`, `ci_upper`, `n_valid` (how many of the `n_boot` resamples produced a usable ICC), and `boot_dist`. Warns if more than 10% of resamples were invalid.

---

ai_calibration *Overall calibration summary for AI system outputs*

Description

Overall calibration summary for AI system outputs

Usage

```
ai_calibration(outcome, confidence, n_bins = 10)
```

Arguments

`outcome`	Binary (0/1) vector of true outcomes.
`confidence`	Numeric vector of AI-stated confidence/probability in [0, 1], same length as `outcome`.
`n_bins`	Integer number of bins for ECE / the calibration curve. Must be >= 2.

Value

A list of class `aiEvalR_calibration`.

`ai_conditional_sem` *Conditional standard error of measurement for AI outputs*

Description

Rather than a single reliability figure, estimates measurement error as a function of the AI's own output level – analogous to conditional SEM in item response theory, this captures the common pattern where an AI system is more consistent on "easy"/typical queries and less consistent near decision boundaries or on unusual inputs.

Usage

```
ai_conditional_sem(responses, n_bins = 5)
```

Arguments

`responses`	Numeric matrix: rows = prompts, columns = repeated occasions.
`n_bins`	Integer number of bins across the observed score range.

Value

A data frame with `score_bin_mid`, `csem`, and `n` per bin.

Examples

```
set.seed(1)
# 40 prompts, 4 repeated occasions each; spread grows with score level
responses <- t(sapply(1:40, function(i) {
  mu <- i / 10
  rnorm(4, mean = mu, sd = 0.1 + mu * 0.05)
}))
ai_conditional_sem(responses, n_bins = 4)
```

ai_dashboard *Integrated AI Evaluation Dashboard*

Description

Combines module-level scores into a single overall quality index and prints a text-based dashboard. Each module score should be pre-computed via the relevant module function and passed in as a single numeric summary in [0, 1] (higher = better); see `Details` for suggested mappings from raw module output to a [0, 1] score.

Usage

```
ai_dashboard(
  reliability,
  fairness,
  robustness,
  calibration,
  hallucination,
  weights = NULL
)
```

Arguments

`reliability`	Numeric in [0, 1] (e.g., ICC or alpha from the reliability module).
`fairness`	Numeric in [0, 1] (e.g., `1 - demographic_parity_diff`).
`robustness`	Numeric in [0, 1] (e.g., `robustness_index` from `stress_test()`).
`calibration`	Numeric in [0, 1] (e.g., `1 - ece`).
`hallucination`	Numeric in [0, 1] (e.g., `1 - hallucination_rate`).
`weights`	Named numeric vector of weights for the five modules above, summing to 1. Defaults to equal weighting.

Details

This module intentionally does NOT attempt to auto-derive [0, 1] scores from raw module output, since the "good" direction and natural scale differ by context. Users should rescale module outputs deliberately before passing them here.

Value

A list of class `aiEvalR_dashboard` with the module scores, `overall` index, and letter `grade`. Printing the object renders a text bar-chart dashboard.

`ai_decision_consistency` *Decision consistency for AI classification/scoring decisions*

Description

When AI outputs are used to make a categorical decision (e.g., pass/fail, flagged/not-flagged) via a threshold, estimates the probability that repeated administrations would yield the same decision, following the classical decision-consistency literature (Livingston & Lewis, 1995 approach, simplified to an empirical resampling estimate here).

Usage

```
ai_decision_consistency(responses, cutpoint)
```

Arguments

`responses`	Numeric matrix: rows = prompts, columns = repeated occasions.
`cutpoint`	Numeric threshold: `response >= cutpoint` is the "positive" decision.

Value

A list with `consistency` (proportion of prompts where all occasions agree on the decision) and `kappa`-style chance-corrected agreement across pairs of occasions.

References

Livingston, S. A., & Lewis, C. (1995). Estimating the consistency and accuracy of classifications based on test scores. *Journal of Educational Measurement, 32*(2), 179-197.

Examples

```
# rows = prompts, cols = repeated occasions
responses <- rbind(
  c(9, 9, 10),  # clearly above a cutpoint of 5 every time
  c(1, 0, 1),   # clearly below
  c(4, 6, 5)    # straddles the cutpoint -> inconsistent decision
)
ai_decision_consistency(responses, cutpoint = 5)
```

ai_dif_mh — *Mantel-Haenszel differential functioning test for AI-scored binary items*

Description

A basic Mantel-Haenszel DIF statistic for a single binary-scored item across two groups, conditioning on a matching (ability/total-score) variable. This is a lightweight helper for a quick outcome-level check – it is **not** a replacement for the `aiDIF` package's IRT-based Differential AI Scoring Bias (DASB) test, which models item parameters jointly across human and AI scoring conditions with proper anchor handling and asymptotic inference. Use `aiDIF::fit_aidif()` for publication-grade item-level AI-scoring-bias analysis; use this function only for a fast two-group screening pass.

Usage

```
ai_dif_mh(item_scores, group, total_score, n_strata = 5)
```

Arguments

`item_scores`	Binary (0/1) vector of item outcomes.
`group`	Grouping factor with exactly two levels (reference vs. focal group).
`total_score`	Numeric vector: total/ability score used to form matched strata.
`n_strata`	Integer. Number of score strata to form. Default 5.

Value

A list with the Mantel-Haenszel odds ratio `mh_or`, its `mh_delta` (ETS delta scale), a chi-square test statistic `chisq` (Mantel-Haenszel, 1 df) and its p-value `p_value`.

ai_dstudy — *D-study: project generalizability/dependability for given facet sizes*

Description

Given the variance components from `ai_generalizability()` (the G-study), computes the generalizability coefficient (relative error, appropriate for rank-order/norm-referenced decisions) and the dependability index Phi (absolute error, appropriate for criterion- referenced decisions, e.g. "is this AI system's score on this case above a fixed bar") for a hypothetical measurement procedure using `n_<facet>` conditions of each facet – which need not match the number of conditions actually observed in the G-study.

Usage

```
ai_dstudy(g, ...)
```

Arguments

g	An aiEvalR_gtheory object from ai_generalizability().
...	Named integers, one per facet used in the G-study (e.g. n_prompt = 3, n_model = 2, n_run = 1), giving the number of conditions of that facet in the measurement procedure being evaluated. Defaults to the number of distinct levels observed in the G-study data for any facet not specified.

Value

A list of class aiEvalR_dstudy with universe_score_variance, relative_error_variance, absolute_error_variance, g_coef, and phi.

Examples

```
set.seed(1)
d <- expand.grid(case = 1:15, prompt = 1:3, model = 1:2, run = 1:2)
d$score <- 5 + rnorm(15)[d$case] + rnorm(3)[d$prompt] * 0.3 +
  rnorm(nrow(d)) * 0.5
g <- ai_generalizability(d)

# dependability at the observed design
ai_dstudy(g, n_prompt = 3, n_model = 2, n_run = 2)

# projected dependability with fewer conditions per facet
ai_dstudy(g, n_prompt = 1, n_model = 1, n_run = 1)
```

---

ai_fairness *Overall fairness summary for AI system outputs*

Description

Reports group-level disparity in AI outcomes (ai_group_disparity()) and, if predicted_probability is supplied, calibration-by-group. For item-level AI-scoring bias in an IRT context (does an AI grader shift item difficulty differently by group), use the companion aiDIF package's fit_aidif() instead – see Details.

Usage

```
ai_fairness(
  outcome,
  group,
  predicted_class = NULL,
  predicted_probability = NULL,
  ...
)
```

Arguments

outcome	Numeric or binary vector of true outcomes/labels.
group	Factor or character vector of group membership.
predicted_class	Optional binary decision vector, for equalized-odds gaps.
predicted_probability	Optional numeric vector of AI-predicted probabilities in [0, 1], for calibration-by-group.
...	Passed to aiDIF::fit_aidif() when human_mle/ai_mle are supplied (requires the aiDIF package): alpha, scale_by, tol, maxit.

Details

aiEvalR's fairness module and the causalfragility-lab/aiDIF package answer different questions and are not meant to be substitutes:

aiEvalR::ai_fairness() Group disparity in AI outcomes at the level of overall outcomes/decisions (demographic parity, equalized odds, calibration-by-group). Works for any outcome type.

aiDIF::fit_aidif() Item-level Differential AI Scoring Bias (DASB) via paired human/AI IRT calibration – whether an AI scoring engine shifts specific item difficulties differently by group, not just whether overall pass rates differ. Requires IRT item parameters, not just outcome vectors.

If aiDIF is installed and item-level MLE lists are supplied via human_mle/ai_mle in ..., ai_fairness() will dispatch to aiDIF::fit_aidif() for that portion of the analysis rather than reimplementing it. Optional alpha, scale_by, tol, maxit in ... are forwarded to fit_aidif() (see that package's documentation); the result is an object of class "aidif" with its own print(), summary(), and plot() methods, plus companion functions aiDIF::ai_effect_summary() and aiDIF::anchor_weights() for further inspection.

Value

A list of class aiEvalR_fairness.

---

ai_generalizability *Multi-facet generalizability (G-)study for AI outputs*

Description

The methodological core of the package: treats the AI system as a measurement instrument, "cases" (the object of measurement – e.g., the underlying task or item being evaluated) as what you want to generalize a score about, and treats prompt formulation, model/model-version, and stochastic run as facets of measurement whose variance is error rather than signal. This generalizes the two-way (prompt x occasion) design to a fully crossed multi-facet design – case x prompt x model x run – via lme4, following classical generalizability theory (Cronbach et al., 1972; Brennan, 2001).

Call ai_generalizability() once per dataset to get variance components (the "G-study"); call ai_dstudy() on the result, as many times as you like, to project generalizability/dependability for different numbers of conditions per facet (the "D-study") without refitting the model.

Usage

```
ai_generalizability(
  data,
  score = "score",
  case = "case",
  facets = c("prompt", "model", "run"),
  include_interactions = TRUE
)
```

Arguments

`data`	A long-format data frame with one row per case x prompt x model x run observation.
`score`	Character. Name of the numeric outcome column.
`case`	Character. Name of the "object of measurement" column (the thing you want a generalizable score about – e.g., a specific task/case ID). Default `"case"`.
`facets`	Character vector of facet column names (conditions of measurement whose variance counts as error). Default `c("prompt", "model", "run")`.
`include_interactions`	Logical. If `TRUE` (default), also estimates `case:facet` two-way interaction variance components (recommended – without these, case-by-facet interaction variance is silently absorbed into the residual and relative-error estimates will be off). Requires design replication to be estimable; if the model fails to converge with interactions, a warning suggests retrying with `include_interactions = FALSE`.

Details

Requires the `lme4` package (listed in Suggests). Interaction terms among facets themselves (e.g. `prompt:model`) are not modeled separately from the residual by default – with typical single- replication designs they usually aren't separately estimable from `case:prompt:model:run`. If your design has replication at that level, extend the formula manually via `lme4::lmer()` directly.

Value

A list of class `aiEvalR_gtheory` with the fitted `lme4` model (`fit`), a `variance_components` data frame, and the `facets` and `case` names used (needed by `ai_dstudy()`).

References

Cronbach, L. J., Gleser, G. C., Nanda, H., & Rajaratnam, N. (1972). *The Dependability of Behavioral Measurements*. Wiley.

Brennan, R. L. (2001). *Generalizability Theory*. Springer.

Examples

```
set.seed(1)
d <- expand.grid(case = 1:15, prompt = 1:3, model = 1:2, run = 1:2)
d$score <- 5 + rnorm(15)[d$case] + rnorm(3)[d$prompt] * 0.3 +
  rnorm(nrow(d)) * 0.5
g <- ai_generalizability(d)
ai_dstudy(g, n_prompt = 3, n_model = 2, n_run = 2)
```

`ai_group_disparity` *Group-level disparity in AI outcomes*

Description

Computes demographic parity difference (mean outcome difference across groups) and, if a binary decision vector is supplied, equalized-odds gaps (difference in true positive / false positive rates across groups). Named "disparity" rather than "bias" deliberately: a between-group difference in outcomes is not automatically evidence of an unfair process – it may reflect legitimate differences, differences in the underlying population, or measurement artifacts. Interpreting a disparity as bias requires additional context this function does not have.

Usage

```
ai_group_disparity(outcome, group, predicted_class = NULL)
```

Arguments

`outcome`	Numeric or binary vector.
`group`	Factor/character grouping vector.
`predicted_class`	Optional binary decision vector (0/1) for equalized odds. Must NOT be a probability – use `ai_fairness()`'s `predicted_probability` for calibration checks instead.

Value

A list with `group_means`, `demographic_parity_diff`, and optionally `equalized_odds`.

ai_internal_consistency *Internal consistency of AI responses across parallel items*

Description

Cronbach's alpha across a set of parallel prompts/items intended to measure the same underlying construct. Requires at least 2 items; alpha is not defined for a single item. Alpha is only a meaningful summary if the items are plausibly unidimensional – this function does not check that assumption.

Usage

```
ai_internal_consistency(responses)
```

Arguments

responses	Numeric matrix or data frame: rows = respondents/runs, columns = items/prompts. Rows with NA are dropped.

Value

A list with alpha and n_items. If total-score variance is zero, alpha is NA with a warning rather than an error.

ai_reliability *Overall reliability summary for AI system outputs*

Description

Convenience wrapper that runs ai_test_retest() and, if items is supplied, ai_internal_consistency(), returning a combined summary. Treats repeated queries to an AI system the way classical test theory treats repeated administrations of a test.

Usage

```
ai_reliability(responses, items = FALSE)
```

Arguments

responses	A numeric matrix or data frame: rows = prompts/items, columns = repeated response occasions (e.g., repeated calls to the same prompt). For consistency measures, columns can instead represent different-but-parallel items.
items	Optional. If responses across columns represent distinct items rather than repeated occasions of the same item, set items = TRUE to also compute internal consistency.

Value

A list of class `aiEvalR_reliability`.

`ai_robustness` *Overall robustness summary for AI system outputs*

Description

Overall robustness summary for AI system outputs

Usage

```
ai_robustness(baseline, perturbed)
```

Arguments

`baseline`	Numeric vector of responses to unperturbed prompts.
`perturbed`	A named list of numeric vectors, one per perturbation type (e.g., `wording`, `spelling`, `order`), each aligned with `baseline` prompt-for-prompt.

Value

A list of class `aiEvalR_robustness`.

`ai_test_retest` *Test-retest reliability of AI responses*

Description

Computes the intraclass correlation coefficient (ICC(2,1), two-way random effects, absolute agreement) across repeated occasions/calls for the same prompt(s), following Shrout & Fleiss (1979).

Usage

```
ai_test_retest(responses)
```

Arguments

`responses`	Numeric matrix: rows = prompts, columns = repeated occasions. Requires at least 2 rows and at least 2 columns. Rows containing any `NA` are dropped (design must be balanced for the classical ANOVA-based ICC used here); a message reports how many rows were dropped.

Value

A list with `icc`, `msb` (between-prompt mean square), `mse` (within-prompt/residual mean square, sometimes denoted MSW in other texts), and `msj` (between-occasion mean square). If the computed ICC falls outside [-1, 1] (can happen with small/unbalanced samples), a warning is issued and the raw value is still returned.

`brier_score` *Brier score*

Description

Brier score

Usage

```
brier_score(outcome, confidence)
```

Arguments

`outcome`	Binary (0/1) vector.
`confidence`	Numeric vector of stated probabilities in [0, 1].

Value

Numeric Brier score (lower is better; 0 = perfect). Computed only over complete cases.

`calibration_curve` *Calibration curve (reliability diagram data)*

Description

Calibration curve (reliability diagram data)

Usage

```
calibration_curve(outcome, confidence, n_bins = 10)
```

Arguments

`outcome`	Binary (0/1) vector.
`confidence`	Numeric vector of stated probabilities in [0, 1].
`n_bins`	Integer number of bins (>= 2).

Value

A data frame with `bin_mid`, `mean_confidence`, `mean_outcome`, and `n` per bin. Bins with zero observations are omitted (not silently treated as zero).

citation_accuracy *Citation accuracy*

Description

For AI outputs that include citations, checks what proportion of cited sources actually exist in a known reference set.

Usage

```
citation_accuracy(cited, known_sources)
```

Arguments

cited	Character vector of citation identifiers produced by the AI (e.g., DOIs, short refs).
known_sources	Character vector of valid/known source identifiers.

Value

A list with `exists_rate` (proportion of citations that are real) and `fabricated` (the citations not found in `known_sources`). Note: existing in `known_sources` confirms the source is real, not that it actually supports the claim it was cited for – that requires a separate entailment check against the cited passage.

ece *Expected Calibration Error*

Description

Expected Calibration Error

Usage

```
ece(outcome, confidence, n_bins = 10)
```

Arguments

outcome	Binary (0/1) vector.
confidence	Numeric vector of stated probabilities in [0, 1].
n_bins	Integer number of bins (>= 2).

Value

Numeric ECE value (lower is better; 0 = perfectly calibrated). The weighting denominator is the count of complete (non-missing, in-range) observations actually assigned to a bin, not the raw input length – so missing/out-of-range values do not silently deflate the estimate.

empirical_output_interval *Empirical output interval for AI numeric outputs*

Description

Empirical (nonparametric) interval spanning the central `conf` proportion of repeated AI outputs for the same query (e.g., from repeated sampling at temperature > 0). Named "empirical output interval" rather than "prediction interval" deliberately: it describes the observed spread of the AI's own repeated outputs, not a calibrated interval with a guaranteed coverage rate for a future true value – those are different claims.

Usage

```
empirical_output_interval(samples, conf = 0.9)
```

Arguments

`samples`	Numeric vector of repeated AI outputs for the same query.
`conf`	Numeric confidence level. Default 0.90.

Value

A list with `lower`, `upper`, `median`.

fairness_dashboard *Fairness dashboard summary (text form)*

Description

Fairness dashboard summary (text form)

Usage

```
fairness_dashboard(fairness_obj)
```

Arguments

`fairness_obj`	An `aiEvalR_fairness` object from `ai_fairness()`.

Value

Invisibly returns `fairness_obj`; prints a summary.

hallucination_rate *Hallucination rate across AI outputs*

Description

Proportion of AI-generated claims flagged as unsupported by a reference source. This function does not itself perform fact-checking (that requires an NLI/retrieval component outside this package's scope) – it aggregates externally-produced, already-adjudicated per-claim verdicts. Treat this as "aggregation of adjudicated hallucination labels," not hallucination detection.

Usage

```
hallucination_rate(verdicts, by = NULL)
```

Arguments

verdicts	Logical or 0/1 vector where TRUE/1 = claim unsupported/hallucinated, one entry per claim, as adjudicated by a human rater or a separate fact-checking system.
by	Optional grouping vector (e.g., by prompt or by document), same length as verdicts, to get per-group rates.

Value

Numeric hallucination rate, or a named vector by group if by is supplied.

lexical_overlap *Lexical overlap between an AI output and a reference source*

Description

Token-overlap-based similarity (Jaccard similarity on token sets) between an AI-generated claim and a reference passage. This is a cheap lexical baseline, NOT a factual-consistency or contradiction-detection measure: negated or opposite-meaning claims can share most of their tokens with the source (e.g. "the treatment reduces mortality" vs. "the treatment does not reduce mortality" score as highly similar here despite opposite meaning). Use an NLI-model-based entailment/ contradiction score for genuine factual-consistency checking; this function is a placeholder with the same interface so that swap can happen without changing calling code.

Usage

```
lexical_overlap(claim, reference)
```

Arguments

claim	Character string: the AI-generated claim.
reference	Character string: the reference/source text.

Value

Numeric lexical overlap score in [0, 1]. NA if either input has no usable tokens.

---

prompt_sensitivity *Prompt sensitivity analysis*

Description

Quantifies how much AI output varies across superficial rewordings of the same underlying prompt (paraphrases), holding semantic content fixed. High variance here signals brittleness rather than genuine uncertainty.

Usage

```
prompt_sensitivity(responses_by_paraphrase)
```

Arguments

responses_by_paraphrase	A list of numeric vectors, one per paraphrase group, each containing responses for the same set of underlying prompts phrased differently.

Value

A list with within_prompt_variance, between_prompt_variance, and sensitivity_ratio (within / between; lower is better; NA if between-prompt variance is zero, e.g. a single underlying prompt). Assumes numeric responses of equal length across paraphrase groups and does not itself estimate uncertainty around the ratio; use ai_bootstrap_reliability()-style resampling if a confidence interval is needed.

---

stress_test *Stress test AI responses under a set of perturbations*

Description

Stress test AI responses under a set of perturbations

Usage

```
stress_test(baseline, perturbed)
```

Arguments

baseline	Numeric vector of unperturbed responses.
perturbed	Named list of numeric vectors of perturbed responses, each the same length as baseline.

Details

robustness_index is 1 - mean(mean_abs_change / sd(baseline)), clipped at 0. It is a convenient descriptive summary for comparing perturbations within one evaluation run, but: (a) it is not guaranteed bounded above by 1 in all cases, (b) it carries no uncertainty estimate, and (c) it depends heavily on sd(baseline), so it is not comparable across AI systems or datasets with different baseline variance. Treat it as descriptive, not as a validated robustness score for reporting or comparison across studies.

Value

A list with per-perturbation mean_abs_change, rmse, and correlation with baseline, plus a descriptive robustness_index (not a validated/calibrated metric – see Details).

Index