

Package ‘decisionfacets’

October 7, 2026

Title Decision Accuracy and Consistency for Rater-Mediated Exams

Version 0.1.0

Description Answers ``would this candidate have passed with a different set of raters?'' for rater-mediated exams such as oral examinations, objective structured clinical examinations and essay scoring. Builds on the many-facet extension of the rating scale model (Andrich, 1978, <[doi:10.1007/BF02293814](https://doi.org/10.1007/BF02293814)>) to compute counterfactual pass probabilities under the observed, an average-severity and a random rater panel, under an explicit decision rule (raw total, fair average or measure), and splits expected misclassification into measurement error and rater assignment, extending item response theory classification accuracy (Lee, 2010, <[doi:10.1111/j.1745-3984.2009.00096.x](https://doi.org/10.1111/j.1745-3984.2009.00096.x)>) to rater effects. Models can be fitted with 'TAM' or a built-in joint maximum likelihood estimator.

License MIT + file LICENSE

URL <https://github.com/edidatasolutions/decisionfacets>,
<https://edidatasolutions.github.io/decisionfacets/>

BugReports <https://github.com/edidatasolutions/decisionfacets/issues>

Encoding UTF-8

Depends R (>= 4.1)

Imports stats, utils

Suggests TAM, knitr, markdown

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Daniel Edi [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0001-5475-819X>>)

Maintainer Daniel Edi <danieledi2026@gmail.com>

Repository CRAN

Date/Publication 2026-10-07 09:50:30 UTC

Contents

df_attribute	2
df_classify	3
df_counterfactual	4
df_cut	5
df_data	6
df_fit	6
df_rater_effects	7
df_simulate	8

Index	10
--------------	-----------

df_attribute	<i>Attribute misclassification to measurement error vs. rater assignment</i>
--------------	--

Description

A decision is a misclassification when it disagrees with the candidate's standing against the intended standard, 'theta >= theta_standard': the measure-scale cut for measure-based rules, or for 'raw_total' the ability at which an average panel's expected total equals the raw cut. For each candidate the probability of misclassification is computed under three panels, and the expected number of wrong decisions is split into:

measurement Error expected with an average-severity panel. It comes from having a finite number of ratings and no rater can remove it.

assignment Observed-panel error minus average-panel error: what the raters actually assigned added (or, if negative, removed). Reported net and as gross harm and gross benefit, because lenient raters can rescue a truly passing borderline candidate while wrongly passing another.

design Random-panel error minus average-panel error: what the random assignment design costs in expectation, before anyone is scored. 'df_design()' will try to reduce this.

Usage

```
df_attribute(counterfactual, components = "severity")
```

Arguments

counterfactual A 'df_counterfactual'.

components Rater effects to separate. Only "severity" is estimable from the current model (no halo or drift terms yet).

Details

With estimated parameters, error probabilities integrate over each candidate's ability posterior. Net components are well recovered in simulation, but the gross harm/benefit split and the false-pass / false-fail split are attenuated by posterior shrinkage; report them from estimates with that caveat.

Value

A 'df_attribution' data frame, one row per candidate: 'person', 'panel', 'pass_observed', 'p_true_pass' (probability the candidate truly meets the standard), 'err_observed', 'err_average', 'err_random', 'assignment' (= err_observed - err_average), 'false_pass' and 'false_fail' (the two parts of err_observed), and 'realized_error' when the counterfactual was computed from known truth. 'summary()' gives the aggregate decomposition.

Examples

```
sim <- df_simulate(n_persons = 200, n_items = 3, n_raters = 6, seed = 1)
cf <- df_counterfactual(sim, df_cut(12, "raw_total"))
df_attribute(cf)
```

df_classify	<i>Classify candidates from their observed scores</i>
-------------	---

Description

Classify candidates from their observed scores

Usage

```
df_classify(object, cut)
```

Arguments

object	A 'df_fit' or 'df_sim'.
cut	A 'df_cut'.

Value

Data frame: 'person', 'panel', 'total', 'raw_cut' (the raw total this candidate's panel had to reach), 'pass'.

Examples

```
sim <- df_simulate(n_persons = 100, n_items = 3, n_raters = 6, seed = 1)
head(df_classify(sim, df_cut(0, "measure")))
```

df_counterfactual	<i>Counterfactual pass probabilities: would this candidate have passed with different raters?</i>
-------------------	---

Description

For each candidate, computes the probability of passing a re-rating under (a) the observed panel, (b) a panel of average-severity raters, and (c) a panel drawn at random from the rater pool, all under the same decision rule. Probabilities are exact (recursive convolution of the model's category probabilities); the only approximation is sampling panels when the pool is too large to enumerate.

Usage

```
df_counterfactual(
  object,
  cut,
  theta = c("auto", "posterior", "point"),
  max_panels = 2000,
  flag_delta = 0.2,
  grid = seq(-6, 6, by = 0.1),
  prior_mean = NULL,
  prior_sd = NULL,
  seed = NULL
)
```

Arguments

object	A 'df_fit' (estimated parameters) or 'df_sim' (true parameters).
cut	A 'df_cut'.
theta	How candidate ability enters: "posterior" integrates over the grid posterior given the candidate's observed ratings (the default for a 'df_fit', so probabilities include measurement error); "point" plugs in 'object\$par\$theta' (the default for a 'df_sim', giving the known truth).
max_panels	Enumerate all rater panels when there are at most this many; otherwise sample this many panels.
flag_delta	Minimum rater advantage (see below) for a flag.
grid	Theta grid for the posterior.
prior_mean, prior_sd	Normal prior for the posterior; default to the fitted population prior when the fit supplies one ('par\$theta_prior'), otherwise the mean and SD of the person estimates.
seed	Optional seed for panel sampling.

Value

A ‘df_counterfactual’ data frame, one row per candidate: ‘person’, ‘panel’, ‘total’, ‘raw_cut’, ‘pass_observed’ (actual decision), ‘p_observed’, ‘p_average’, ‘p_random’, ‘p_min’, ‘p_max’ (worst and best panel in the pool), ‘delta’ (= p_observed - p_random), ‘advantage’, ‘direction’ and ‘rater_dependent’.

‘advantage’ is how much the assigned panel pushed the candidate toward the outcome they actually received: ‘delta’ for a pass, ‘-delta’ for a fail. Equivalently, it is the increase in the probability that a re-rating would reverse the decision when the observed panel is swapped for a random one; decision reversals from measurement error alone cancel out. ‘rater_dependent’ is ‘advantage >= flag_delta’, and ‘direction’ labels flagged cases “lenient_panel_pass” (board’s false-pass exposure) or “harsh_panel_fail” (the appeal case).

Examples

```
sim <- df_simulate(n_persons = 200, n_items = 3, n_raters = 6, seed = 1)
fit <- df_fit(sim$data, engine = "jmlr")
cf <- df_counterfactual(fit, df_cut(12, "raw_total"))
cf                                     # most rater-dependent candidates first
summary(cf)
# Known truth: the same analysis with the true parameters
summary(df_counterfactual(sim, df_cut(12, "raw_total")))
```

df_cut

*Define a pass/fail cut under an explicit decision rule***Description**

The decision rule determines how rater severity can reach the decision:

- ‘**raw_total**’ Pass if the summed observed ratings reach ‘value’. Severity passes straight through to the decision.
- ‘**measure**’ Pass if the severity-adjusted Rasch measure (logits) reaches ‘value’. Severity is modeled out; only its effect on measurement precision remains.
- ‘**fair_average**’ Pass if the FACETS-style fair average (expected mean rating per cell for an average rater) reaches ‘value’. It is monotone in the measure, so it behaves like ‘measure’ with a transformed cut.

Usage

```
df_cut(value, decision_rule = c("raw_total", "fair_average", "measure"))
```

Arguments

- value The cut score on the scale implied by ‘decision_rule’.
- decision_rule One of “raw_total”, “fair_average”, “measure”.

Value

A ‘df_cut’ object (a list with ‘value’ and ‘decision_rule’).

Examples

```
df_cut(16, "raw_total")      # pass if the summed ratings reach 16
df_cut(2, "fair_average")    # pass if the fair average reaches 2
df_cut(0.25, "measure")      # pass if the Rasch measure reaches 0.25 logits
```

df_data	<i>Standardize rater-mediated score data</i>
---------	--

Description

Standardize rater-mediated score data

Usage

```
df_data(x, person = "person", item = "item", rater = "rater", score = "score")
```

Arguments

x A data frame in long format, one row per person x item x rater score.
person, item, rater, score
 Column names in 'x'.

Value

A 'df_data' data frame with character ids and 0-based integer scores; attribute 'K' is the highest category.

Examples

```
raw <- data.frame(cand = c("A", "A", "B", "B"), task = "T1",
                  examiner = c("R1", "R2", "R1", "R2"), rating = c(3, 4, 2, 2))
df_data(raw, person = "cand", item = "task", rater = "examiner", score = "rating")
```

df_fit	<i>Fit a many-facet Rasch rating scale model</i>
--------	--

Description

Fit a many-facet Rasch rating scale model

Usage

```
df_fit(data, engine = NULL, max_iter = 1000, tol = NULL)
```

Arguments

data	A 'df_data' object.
engine	"tam" (marginal ML via 'TAM::tam.mml.mfr', the default when TAM is installed) or "jmle" (built-in joint maximum likelihood, the FACETS approach).
max_iter, tol	Convergence controls; 'tol = NULL' uses each engine's default (1e-4 for TAM, 1e-6 for JMLE).

Details

Both engines report parameters in the same parameterization: item difficulties and rater severities centered at 0, thresholds centered at 0, and person measures on the resulting logit scale. For TAM, 'par\$theta' holds EAPs.

JMLE person measures are clamped to [-7, 7], so extreme scores get a finite but arbitrary measure. JMLE's known small-sample spread inflation is not corrected.

Value

A 'df_fit' object: '\$data', '\$par' (named 'theta', 'delta', 'lambda', 'tau'; for TAM also 'theta_prior', the fitted population mean and SD), '\$engine', '\$converged', '\$iterations', and for TAM the fitted '\$model'.

Examples

```
sim <- df_simulate(n_persons = 200, n_items = 3, n_raters = 6, seed = 1)
fit <- df_fit(sim$data, engine = "jmle")
cor(fit$par$lambda, sim$par$lambda[names(fit$par$lambda)])

if (requireNamespace("TAM", quietly = TRUE)) {
  fit_tam <- df_fit(sim$data, engine = "tam")
  fit_tam$par$tau
}
```

df_rater_effects	<i>Rater severity estimates</i>
------------------	---------------------------------

Description

Rater severity estimates

Usage

```
df_rater_effects(object)
```

Arguments

object	A 'df_fit' or 'df_sim'.
--------	-------------------------

Value

A data frame of rater ids and severities (logits, centered at 0; positive = harsher).

Examples

```
sim <- df_simulate(n_persons = 100, n_items = 3, n_raters = 6, seed = 1)
df_rater_effects(sim)
```

df_simulate

Simulate a rater-mediated administration with known truth

Description

Each candidate is scored on every item by a panel of ‘raters_per_person’ raters drawn at random from the pool.

Usage

```
df_simulate(
  n_persons = 500,
  n_items = 4,
  n_raters = 12,
  raters_per_person = 2,
  n_cat = 5,
  theta_mean = 0,
  theta_sd = 1,
  item_sd = 0.5,
  severity_sd = 0.5,
  tau = NULL,
  seed = NULL
)
```

Arguments

n_persons, n_items, n_raters

Facet sizes.

raters_per_person

Panel size per candidate.

n_cat

Number of score categories (scores 0..n_cat-1).

theta_mean, theta_sd

Candidate ability distribution.

item_sd, severity_sd

SDs of item difficulty and rater severity (both centered).

tau

Category thresholds; defaults to equally spaced on [-1.5, 1.5].

seed

Optional RNG seed.

Value

A 'df_sim' object: '\$data' (a 'df_data') and '\$par', the true parameters ('theta', 'delta', 'lambda', 'tau', all named).

Examples

```
sim <- df_simulate(n_persons = 100, n_items = 3, n_raters = 6, seed = 1)
head(sim$data)
sim$par$lambda  # true rater severities
```

Index

`df_attribute`, [2](#)
`df_classify`, [3](#)
`df_counterfactual`, [4](#)
`df_cut`, [5](#)
`df_data`, [6](#)
`df_fit`, [6](#)
`df_rater_effects`, [7](#)
`df_simulate`, [8](#)