

Package ‘mixedsubjectsirt’

June 25, 2026

Title Item Response Theory Calibration with a Mixed Subjects Design

Version 1.0.0

Description Integrates large language model generated item responses into psychometric calibration studies through a mixed-subjects design for unidimensional two-parameter and one-parameter logistic item response theory models. Human pilot responses are augmented with model-generated responses using a prediction-powered inference estimator (Angelopoulos, Bates, Fannjiang, Jordan and Zrnic (2023) <[doi:10.1126/science.adi6000](https://doi.org/10.1126/science.adi6000)>; Angelopoulos, Duchi and Zrnic (2023) <[doi:10.48550/arXiv.2311.01453](https://doi.org/10.48550/arXiv.2311.01453)>) adapted to marginal maximum-likelihood estimation, following the mixed-subjects design of Broska, Howes and van Loon (2025) <[doi:10.1177/00491241251326865](https://doi.org/10.1177/00491241251326865)>. The estimator is anchored to the human responses and is asymptotically unbiased for the human item parameters at any tuning weight; the weight on the synthetic responses is chosen to minimize propagated ability-score risk, down-weighting uninformative or biased generated responses. Louis-corrected sandwich standard errors, ability scoring, cross-fitted tuning, and scale linking are also provided.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Imports mirt, rmutl

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://kintkanopka.com/mixedsubjectsirt/>,
<https://github.com/kintkanopka/mixedsubjectsirt>

BugReports <https://github.com/kintkanopka/mixedsubjectsirt/issues>

NeedsCompilation no

Author Klint Kanopka [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-3196-9538>>)

Maintainer Klint Kanopka <klint.kanopka@nyu.edu>

Repository CRAN

Date/Publication 2026-06-25 15:50:09 UTC

Contents

ability_gradient	3
ability_gradient_1pl	3
ability_risk	4
ability_risk_1pl	5
diagnose_lambda_grid	6
fit_1pl	7
fit_2pl	8
fit_mixed_subjects	9
fit_mixed_subjects_1pl	10
fit_mixed_subjects_from_quadrature	12
fit_mixed_subjects_iterative	13
fit_mixed_subjects_mml	15
fit_mixed_subjects_mml_1pl	17
fit_mixed_subjects_split	19
link_item_parameters	21
make_quadrature	22
mixed_subjects_loss	22
mixed_subjects_quadrature	23
posterior_weights_2pl	24
score_theta	25
simulate_2pl	26
summarize_expected_counts	27
tune_lambda_ability_risk	27
tune_lambda_ability_risk_1pl	30
tune_lambda_ability_risk_crossfit	32
tune_lambda_ability_risk_item	34
tune_lambda_ppi_score	36
tune_lambda_ppi_score_1pl	37
tune_lambda_ppi_score_item	38
vcov_mixed_subjects	40
vcov_mixed_subjects_1pl	40
vcov_mixed_subjects_mml	41

Index

43

ability_gradient *Gradient of ML ability scores with respect to item parameters*

Description

Computes the implicit derivative of bounded maximum-likelihood ability scores with respect to 2PL item parameters. The column order is all discriminations followed by all intercepts.

Usage

```
ability_gradient(resp, item_pars, theta = NULL, bounds = c(-6, 6), eps = 1e-10)
```

Arguments

resp	Response matrix with rows for subjects and columns for items.
item_pars	Item parameters in slope-intercept form, or a "mixedsubjects_fit" object.
theta	Optional precomputed ability estimates. If omitted, <code>score_theta()</code> is used.
bounds	Bounds passed to <code>score_theta()</code> when theta is omitted.
eps	Tolerance used to mark near-zero test information as undefined.

Value

A matrix with one row per response pattern and one column per item parameter.

Examples

```
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- matrix(c(1, 0, 0, 1), nrow = 2, byrow = TRUE)
ability_gradient(resp, pars)
```

ability_gradient_1pl *Gradient of ML ability scores w.r.t. 1PL item parameters*

Description

Computes the implicit derivative of bounded maximum-likelihood ability scores with respect to the 1PL parameters (a_shared, d_1, ..., d_J).

Usage

```
ability_gradient_1pl(
  resp,
  item_pars,
  theta = NULL,
  bounds = c(-6, 6),
  eps = 1e-10
)
```

Arguments

resp	Response matrix.
item_pars	Item parameters with all a equal (IPL), or a "mixedsubjects_1pl_fit" object.
theta	Optional precomputed ability estimates.
bounds	Bounds passed to score_theta() .
eps	Tolerance for near-zero test information.

Details

The gradient for the shared discrimination is the sum of the per-item discrimination gradients: $da_shared = \sum_j da_j$ (chain rule via the constraint $a_j = a_shared$).

Value

A matrix with one row per response pattern and $J + 1$ columns (a_shared , then one column per item's d_j).

ability_risk	<i>Propagated ability risk from item-parameter uncertainty</i>
--------------	--

Description

Computes g_i' Sigma g_i for each response pattern, where g_i is the gradient of the ability estimate with respect to item parameters. If $theta_true$ is supplied, the returned total risk also includes squared ability estimation error.

Usage

```
ability_risk(
  resp,
  fit_or_pars,
  vcov = NULL,
  theta_true = NULL,
  bounds = c(-6, 6)
)
```

Arguments

resp	Target response matrix.
fit_or_pars	A "mixedsubjects_fit" object or item-parameter data frame.
vcov	Optional covariance matrix. Required when <code>fit_or_pars</code> is item parameters rather than a fitted mixed-subjects object.
theta_true	Optional true theta values for simulation studies.
bounds	Bounds passed to score_theta() .

Value

A list with summary and per-pattern details.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- simulate_2pl(rnorm(30), pars)
Sigma <- diag(0.01, 4)
ability_risk(resp, pars, vcov = Sigma)$summary
```

ability_risk_1pl	<i>Propagated ability risk for a IPL fit</i>
------------------	--

Description

Computes $\mathbf{g}_i' \Sigma_{1pl} \mathbf{g}_i$ for each response pattern, where \mathbf{g}_i is the $(J+1)$ -dimensional gradient of the ability estimate with respect to $(a_{\text{shared}}, d_1, \dots, d_J)$ and Σ_{1pl} is the sandwich covariance from [vcov_mixed_subjects_1pl\(\)](#).

Usage

```
ability_risk_1pl(
  resp,
  fit_or_pars,
  vcov = NULL,
  theta_true = NULL,
  bounds = c(-6, 6)
)
```

Arguments

resp	Target response matrix.
fit_or_pars	A "mixedsubjects_1pl_fit" object or item-parameter data frame.
vcov	Optional $(J+1) \times (J+1)$ covariance matrix. Required when fit_or_pars is not a fitted object.
theta_true	Optional true theta values for simulation studies.
bounds	Bounds passed to score_theta() .

Value

A list with summary and per-pattern details, the same structure as [ability_risk\(\)](#).

diagnose_lambda_grid *Diagnose lambda values over a grid*

Description

Fits `fit_mixed_subjects()` or `fit_mixed_subjects_split()` over a set of candidate lambda values. The returned summary reports the fitted mixed-subjects objective and the observed human expected-count loss for each candidate. This is a sensitivity diagnostic, not a valid tuning rule.

Usage

```
diagnose_lambda_grid(
  lambda_grid,
  observed,
  predicted,
  generated,
  split = FALSE,
  ...
)
```

Arguments

`lambda_grid` Numeric vector of lambda values in $[0, 1]$.
`observed, predicted, generated` Response matrices passed to `fit_mixed_subjects()`.
`split` Logical; if TRUE, call `fit_mixed_subjects_split()`.
`...` Additional arguments passed to the selected fitting function.

Value

A list with `summary`, `lowest_observed_loss_lambda`, and all fitted model objects.

Examples

```
set.seed(3)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(30), pars)
predicted <- observed
generated <- simulate_2pl(rnorm(80), pars)
tuned <- diagnose_lambda_grid(
  c(0, 0.5),
  observed, predicted, generated,
  initial_pars = pars, n_quad = 5, control = list(maxit = 30)
)
tuned$summary
```

`fit_1pl`*Fit a 1PL (one-parameter logistic) model*

Description

Estimates a shared discrimination parameter a (equal across all items) and per-item intercepts d_j by maximizing the IRT marginal likelihood under a standard-normal ability prior using L-BFGS-B.

Usage

```
fit_1pl(  
  resp,  
  n_quad = 31,  
  initial_pars = NULL,  
  quadrature = NULL,  
  slope_lower = 1e-04,  
  slope_upper = NULL,  
  control = list(maxit = 500)  
)
```

Arguments

<code>resp</code>	Binary response matrix.
<code>n_quad</code>	Number of standard-normal quadrature nodes.
<code>initial_pars</code>	Optional starting item parameters (data frame with a and d columns). If omitted, $a = 1$ and $d_j = \text{qlogis}(p_j)$ where p_j is the observed proportion correct for item j .
<code>quadrature</code>	Optional quadrature grid.
<code>slope_lower, slope_upper</code>	Bounds on the shared discrimination.
<code>control</code>	Control list passed to <code>stats::optim()</code> .

Details

The response probability is $P(x_j = 1 \mid \theta) = \text{plogis}(a * \theta + d_j)$. The parameter vector has length $J + 1$: one shared discrimination followed by J per-item intercepts.

Value

A list with `pars` (item parameter data frame with all a equal), `par` (the raw parameter vector), and optimizer details.

Examples

```
set.seed(1)
pars <- data.frame(a = 1, d = c(-0.5, 0, 0.5))
resp <- simulate_2pl(rnorm(60), pars)
fit <- fit_1pl(resp, n_quad = 7)
fit$pars
```

fit_2pl

Fit a unidimensional 2PL IRT model

Description

Fits a two-parameter logistic model with `mirt` and returns item parameters in slope-intercept form. The response probability is $\text{plogis}(d + a * \text{theta})$, where a is the discrimination and d is the intercept. Difficulty is returned as $b = -d / a$.

Usage

```
fit_2pl(resp, technical = list(NCYCLES = 1000), verbose = FALSE, ...)
```

Arguments

<code>resp</code>	A numeric item response matrix with rows for subjects and columns for items. Values must be binary 0/1; NA is allowed.
<code>technical</code>	A list passed to the <code>technical</code> argument of <code>mirt::mirt()</code> .
<code>verbose</code>	Logical; passed to <code>mirt::mirt()</code> .
<code>...</code>	Additional arguments passed to <code>mirt::mirt()</code> .

Value

A list with `pars`, a data frame containing `item`, `a`, `d`, and `b`, and `model`, the fitted `mirt` model.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9, 1.1, 0.8), d = c(0, 0.5, -0.5, 0.2, -0.3))
resp <- simulate_2pl(rnorm(500), pars)
fit <- fit_2pl(resp)
fit$pars
```

fit_mixed_subjects *Fit a mixed-subjects 2PL calibration*

Description

Fits item parameters using observed human responses, paired LLM responses/predictions for those same subjects, and generated or unlabeled LLM responses. This implements the expected-count objective

Usage

```
fit_mixed_subjects(
  observed,
  predicted,
  generated,
  lambda = 1,
  n_quad = 31,
  initial_pars = NULL,
  quadrature = NULL,
  common_predicted_weights = TRUE,
  paired_missing = c("match_observed", "allow"),
  slope_lower = 1e-04,
  slope_upper = NULL,
  control = list(maxit = 500),
  ...
)
```

Arguments

observed	Human response matrix, with rows for subjects and columns for items. Values must be binary when <code>initial_pars</code> is omitted.
predicted	Binary LLM responses (0/1) for the same rows and items as observed. Probabilities are not accepted: fractional values are not a valid likelihood input for the marginal IRT objective and break the PPI correction, so sample binary responses from any probabilities first (e.g. <code>rbinom</code>).
generated	Binary generated or unlabeled LLM responses (0/1) for the same item columns. Probabilities are not accepted (see <code>predicted</code>).
lambda	Power-tuning parameter in $[0, 1]$.
n_quad	Number of standard-normal quadrature nodes.
initial_pars	Optional starting item parameters. If omitted, a 2PL model is fit to observed.
quadrature	Optional quadrature grid with theta and weight columns.
common_predicted_weights	Logical; if TRUE, reuse the observed human posterior weights for predicted.

paired_missing	How to handle missingness when <code>common_predicted_weights = TRUE</code> . The default, "match_observed", requires observed and predicted to have the same missingness pattern so the paired LLM correction is evaluated only where a human label is present. Use "allow" only for explicit sensitivity analyses.
slope_lower	Lower bound for discrimination parameters during optimization. Use NULL for no lower bound.
slope_upper	Upper bound for discrimination parameters during optimization. Use NULL (the default) for no upper bound. Setting a finite bound (e.g. 4) can stabilize the frozen expected-count fit when the LLM parameters differ substantially from the human parameters.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_2pl()</code> when <code>initial_pars</code> is omitted.

Details

$L_{\text{human}} + \lambda * (L_{\text{generated}} - L_{\text{paired_llm}})$.

By default the paired LLM responses reuse the posterior quadrature weights from the observed human responses. This keeps the paired human and LLM terms on the same latent covariate distribution, which is the closest analog to prediction-powered inference with paired labels.

Value

An object of class "mixedsubjects_fit" with fitted `item_pars`, optimizer details, quadrature summaries, and input settings.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
predicted <- observed
generated <- simulate_2pl(rnorm(100), pars)
fit <- fit_mixed_subjects(
  observed, predicted, generated,
  lambda = 0.5, initial_pars = pars, n_quad = 7,
  control = list(maxit = 50)
)
fit$item_pars
```

fit_mixed_subjects_1pl

Fit a mixed-subjects IPL calibration (frozen expected-count)

Description

Analogous to `fit_mixed_subjects()` but estimates a shared discrimination parameter a across all items (IPL model). Posterior quadrature weights are frozen at the initial parameter estimates.

Usage

```
fit_mixed_subjects_1pl(
  observed,
  predicted,
  generated,
  lambda = 1,
  n_quad = 31,
  initial_pars = NULL,
  quadrature = NULL,
  common_predicted_weights = TRUE,
  slope_lower = 1e-04,
  slope_upper = NULL,
  control = list(maxit = 500),
  ...
)
```

Arguments

observed	Human response matrix, with rows for subjects and columns for items. Values must be binary when <code>initial_pars</code> is omitted.
predicted	Binary LLM responses (0/1) for the same rows and items as observed. Probabilities are not accepted: fractional values are not a valid likelihood input for the marginal IRT objective and break the PPI correction, so sample binary responses from any probabilities first (e.g. <code>rbinom</code>).
generated	Binary generated or unlabeled LLM responses (0/1) for the same item columns. Probabilities are not accepted (see <code>predicted</code>).
lambda	Power-tuning parameter in $[0, 1]$.
n_quad	Number of standard-normal quadrature nodes.
initial_pars	Optional starting item parameters. If omitted, a 2PL model is fit to observed.
quadrature	Optional quadrature grid with theta and weight columns.
common_predicted_weights	Logical; if TRUE, reuse the observed human posterior weights for predicted.
slope_lower	Lower bound for discrimination parameters during optimization. Use NULL for no lower bound.
slope_upper	Upper bound for discrimination parameters during optimization. Use NULL (the default) for no upper bound. Setting a finite bound (e.g. 4) can stabilize the frozen expected-count fit when the LLM parameters differ substantially from the human parameters.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_1pl()</code> when <code>initial_pars</code> is omitted.

Value

An object of class `c("mixedsubjects_1pl_fit", "mixedsubjects_fit")`.

See Also

[fit_mixed_subjects_mml_1pl\(\)](#) for the marginal-likelihood version; [fit_mixed_subjects\(\)](#) for the 2PL version.

Examples

```
set.seed(1)
pars <- data.frame(a = 1, d = c(-0.5, 0, 0.5))
observed <- simulate_2pl(rnorm(40), pars)
generated <- simulate_2pl(rnorm(100), pars)
fit <- fit_mixed_subjects_1pl(
  observed, observed, generated,
  lambda = 0.5, initial_pars = pars, n_quad = 7,
  control = list(maxit = 50)
)
fit$item_pars
```

```
fit_mixed_subjects_from_quadrature
```

Fit from precomputed quadrature summaries

Description

Fits the mixed-subjects 2PL objective from quadrature/count summaries rather than raw response matrices. This lower-level interface is useful when the human, paired LLM, and generated LLM summaries have already been linked onto a common scale outside the package.

Usage

```
fit_mixed_subjects_from_quadrature(
  q_observed,
  q_predicted,
  q_generated,
  lambda = 1,
  initial_pars = NULL,
  slope_lower = 1e-04,
  slope_upper = NULL,
  control = list(maxit = 500)
)
```

Arguments

q_observed	Quadrature summary for observed human responses. Usually returned by mixed_subjects_quadrature but a raw counts object returned by summarize_expected_counts() is also accepted.
q_predicted	Quadrature summary for paired LLM responses/predictions on the labeled human rows.

q_generated	Quadrature summary for generated or unlabeled LLM responses.
lambda	Power-tuning parameter in $[0, 1]$.
initial_pars	Starting item parameters in slope-intercept form. If omitted, q_observed\$irt_pars is used when available.
slope_lower	Lower bound for discrimination parameters during optimization. Use NULL for no lower bound.
slope_upper	Upper bound for discrimination parameters during optimization. Use NULL (the default) for no upper bound.
control	Control list passed to <code>stats::optim()</code> .

Value

An object of class "mixedsubjects_fit".

Examples

```

pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- matrix(c(1, 0, 0, 1), nrow = 2, byrow = TRUE)
q <- mixed_subjects_quadrature(resp, item_pars = pars, N_quad = 5)
fit_mixed_subjects_from_quadrature(q, q, q, lambda = 0.5)$item_pars

```

fit_mixed_subjects_iterative

Fit a mixed-subjects 2PL calibration with iterative EM

Description

Extends `fit_mixed_subjects()` by iterating the E-step and M-step until convergence rather than fixing posterior quadrature weights at the initial parameter estimates. At every iteration the posterior weights for all three datasets (observed, predicted, generated) are recomputed using the same current item parameters. This keeps the posteriors internally consistent and avoids the asymmetry between L_{pred} and L_{gen} that arises when frozen human-MLE weights are applied to LLM data with different item parameters.

Usage

```

fit_mixed_subjects_iterative(
  observed,
  predicted,
  generated,
  lambda = 1,
  n_quad = 31,
  initial_pars = NULL,
  quadrature = NULL,
  common_predicted_weights = TRUE,
  paired_missing = c("match_observed", "allow"),

```

```

    slope_lower = 1e-04,
    slope_upper = NULL,
    tol = 1e-04,
    em_maxit = 30,
    control = list(maxit = 200),
    ...
)

```

Arguments

observed	Human response matrix, with rows for subjects and columns for items. Values must be binary when <code>initial_pars</code> is omitted.
predicted	Binary LLM responses (0/1) for the same rows and items as observed. Probabilities are not accepted: fractional values are not a valid likelihood input for the marginal IRT objective and break the PPI correction, so sample binary responses from any probabilities first (e.g. <code>rbinom</code>).
generated	Binary generated or unlabeled LLM responses (0/1) for the same item columns. Probabilities are not accepted (see <code>predicted</code>).
lambda	Power-tuning parameter in $[\emptyset, 1]$.
n_quad	Number of standard-normal quadrature nodes.
initial_pars	Optional starting item parameters. If omitted, a 2PL model is fit to <code>observed</code> .
quadrature	Optional quadrature grid with <code>theta</code> and <code>weight</code> columns.
common_predicted_weights	Logical; if TRUE, reuse the observed human posterior weights for <code>predicted</code> .
paired_missing	How to handle missingness when <code>common_predicted_weights = TRUE</code> . The default, "match_observed", requires observed and predicted to have the same missingness pattern so the paired LLM correction is evaluated only where a human label is present. Use "allow" only for explicit sensitivity analyses.
slope_lower	Lower bound for discrimination parameters during optimization. Use NULL for no lower bound.
slope_upper	Upper bound on discrimination parameters. Strongly recommended when <code>lambda > 0</code> — the iterative EM updates posteriors at each step, and without an upper bound the gradient asymmetry between <code>L_pred</code> and <code>L_gen</code> can compound across iterations, driving discrimination estimates to extreme values. A typical choice is <code>slope_upper = 4</code> or <code>slope_upper = 6</code> .
tol	Convergence tolerance: maximum absolute change in any parameter across an EM iteration.
em_maxit	Maximum number of EM iterations.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_2pl()</code> when <code>initial_pars</code> is omitted.

Details

Note on lambda selection. This function accepts a fixed `lambda`. For psychometric applications where accurate ability scoring is the goal, select `lambda` with `tune_lambda_ability_risk()`

rather than `tune_lambda_ppi_score()`. The PPI++ score objective minimizes the trace of the item-parameter covariance matrix; `tune_lambda_ability_risk()` minimizes the propagated ability-score risk $g' \Sigma g$, which is the quantity that matters for downstream test scoring.

Value

An object of class "mixedsubjects_fit" with the standard fields plus `em_iterations` (number of EM cycles completed) and `em_converged` (logical).

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
predicted <- observed
generated <- simulate_2pl(rnorm(100), pars)
fit <- fit_mixed_subjects_iterative(
  observed, predicted, generated,
  lambda = 0.5, initial_pars = pars, n_quad = 7,
  control = list(maxit = 50), em_maxit = 5
)
fit$item_pars
```

fit_mixed_subjects_mml

Fit a mixed-subjects 2PL calibration via marginal maximum likelihood

Description

Estimates item parameters using the true IRT marginal likelihood for all three loss terms. Unlike `fit_mixed_subjects()`, which freezes posterior quadrature weights at the initial parameter estimates before optimizing, this function recomputes posterior weights at every gradient evaluation. This eliminates the gradient asymmetry that causes `fit_mixed_subjects()` to converge to false minima at inflated discrimination values when LLM item parameters differ from human parameters.

Usage

```
fit_mixed_subjects_mml(
  observed,
  predicted,
  generated,
  lambda = 1,
  n_quad = 31,
  initial_pars = NULL,
  quadrature = NULL,
  mml_pred_weights = c("own", "human"),
  slope_lower = 1e-04,
```

```

slope_upper = NULL,
control = list(maxit = 500),
...
)

```

Arguments

observed	Human response matrix, with rows for subjects and columns for items. Values must be binary when <code>initial_pars</code> is omitted.
predicted	Binary LLM responses (0/1) for the same rows and items as observed. Probabilities are not accepted: fractional values are not a valid likelihood input for the marginal IRT objective and break the PPI correction, so sample binary responses from any probabilities first (e.g. <code>rbinom</code>).
generated	Binary generated or unlabeled LLM responses (0/1) for the same item columns. Probabilities are not accepted (see <code>predicted</code>).
lambda	Power-tuning parameter in $[0, 1]$.
n_quad	Number of standard-normal quadrature nodes.
initial_pars	Optional starting item parameters. If omitted, a 2PL model is fit to <code>observed</code> .
quadrature	Optional quadrature grid with <code>theta</code> and <code>weight</code> columns.
mml_pred_weights	How to compute posteriors for the paired <code>predicted</code> term. "own" uses posteriors from the <code>predicted</code> responses; "human" uses posteriors from the observed human responses. See <code>Details</code> .
slope_lower	Lower bound for discrimination parameters during optimization. Use <code>NULL</code> for no lower bound.
slope_upper	Upper bound on discrimination parameters. Unlike <code>fit_mixed_subjects()</code> , this function should not require capping for well-posed problems because the true marginal objective has no false minimum at large discrimination.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_2pl()</code> when <code>initial_pars</code> is omitted.

Details

Why it matters for lambda selection. With the frozen expected-count implementation, the gradient of `L_pred` uses concentrated human posteriors while `L_gen` uses diffuse LLM posteriors, making `grad(L_pred) >> grad(L_gen)` and systematically pushing discriminations upward at any $\lambda > 0$. In the marginal-MML formulation all three terms use their own current-parameter posteriors, so the asymmetry is absent at the true optimum. As a result `tune_lambda_ability_risk()` selects $\lambda > 0$ whenever the LLM predictions are genuinely informative (e.g. `predicted = observed`), rather than collapsing to $\lambda = 0$ for all misaligned LLMs.

`mml_pred_weights`.

"own" (**default**) `L_pred` uses posteriors computed from the *predicted* response matrix at the current parameter values. All three terms are true marginal likelihoods; objective and gradient are internally consistent. Recommended for most applications and required for `vcov_mixed_subjects_mml()` to produce the fully correct Louis-formula bread.

"human" `L_pred` uses posteriors computed from the *observed* (human) response matrix, frozen at `initial_pars`. This is a **fixed-nuisance Q-function**: the predicted term is treated as a frozen expected-count lower bound rather than a true marginal likelihood. Objective and gradient are mutually consistent (both use the same frozen posteriors) so L-BFGS-B converges correctly. Useful when strong ability-level pairing is needed. Note that `vcov_mixed_subjects_mml()` applies Louis' formula to the stored fixed posteriors, which is approximately correct when `initial_pars` is close to `conv_pars`.

Per-item lambda (vector lambda). When `lambda` is a length-`n_items` vector rather than a scalar, `fit_mixed_subjects_mml` switches to a **frozen Q-function** objective: expected-count counts are computed once from `initial_pars` and held fixed during L-BFGS-B, with item `j`'s counts weighted by `lambda[j]`. This is a consistent (objective, gradient) pair but is *not* the full marginal-MML objective — it is a frozen expected-count approximation analogous to `fit_mixed_subjects()`. Per-item `lambda` values obtained from `tune_lambda_ability_risk_item()` assign `lambda_j` near 0 to items where the LLM correction is harmful, containing the frozen-posterior gradient asymmetry. Document per-item `lambda` results as approximate.

Value

An object of class "mixedsubjects_fit" with the same structure as `fit_mixed_subjects()`. For **scalar** `lambda` fits, the quadrature summaries store posteriors at the converged parameters, and `stats::vcov()` dispatches automatically to `vcov_mixed_subjects_mml()` to compute the Louis-corrected marginal sandwich covariance. Calling `vcov_mixed_subjects()` directly bypasses the Louis correction. For **vector** `lambda` fits, the summaries store the frozen posteriors used during optimization, and `stats::vcov()` dispatches to `vcov_mixed_subjects()` (EM bread) for consistency with the frozen Q-function objective.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
generated <- simulate_2pl(rnorm(100), pars)
fit <- fit_mixed_subjects_mml(
  observed, observed, generated,
  lambda = 0.5, initial_pars = pars, n_quad = 7,
  control = list(maxit = 100)
)
fit$item_pars
```

```
fit_mixed_subjects_mml_1pl
```

Fit a mixed-subjects IPL calibration via marginal maximum likelihood

Description

Analogous to `fit_mixed_subjects_mml()` but estimates a shared discrimination parameter `a` across all items (IPL model). Posteriors are recomputed at every gradient evaluation — no frozen-posterior gradient asymmetry.

Usage

```
fit_mixed_subjects_mml_1pl(
  observed,
  predicted,
  generated,
  lambda = 1,
  n_quad = 31,
  initial_pars = NULL,
  quadrature = NULL,
  mml_pred_weights = c("own", "human"),
  slope_lower = 1e-04,
  slope_upper = NULL,
  control = list(maxit = 500),
  ...
)
```

Arguments

observed	Human response matrix, with rows for subjects and columns for items. Values must be binary when <code>initial_pars</code> is omitted.
predicted	Binary LLM responses (0/1) for the same rows and items as observed. Probabilities are not accepted: fractional values are not a valid likelihood input for the marginal IRT objective and break the PPI correction, so sample binary responses from any probabilities first (e.g. <code>rbinom</code>).
generated	Binary generated or unlabeled LLM responses (0/1) for the same item columns. Probabilities are not accepted (see <code>predicted</code>).
lambda	Power-tuning parameter in $[0, 1]$.
n_quad	Number of standard-normal quadrature nodes.
initial_pars	Optional starting item parameters. If omitted, a 2PL model is fit to observed.
quadrature	Optional quadrature grid with <code>theta</code> and <code>weight</code> columns.
mml_pred_weights	How to compute posteriors for the paired predicted term. "own" uses posteriors from the predicted responses; "human" uses posteriors from the observed human responses. See Details.
slope_lower	Lower bound for discrimination parameters during optimization. Use <code>NULL</code> for no lower bound.
slope_upper	Upper bound on discrimination parameters. Unlike <code>fit_mixed_subjects()</code> , this function should not require capping for well-posed problems because the true marginal objective has no false minimum at large discrimination.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_1pl()</code> when <code>initial_pars</code> is omitted.

Details

Only scalar `lambda` is supported; per-item `lambda` is not meaningful for the 1PL because the discrimination is shared across items.

Value

An object of class `c("mixedsubjects_1pl_fit", "mixedsubjects_fit")`.

See Also

[fit_mixed_subjects_1pl\(\)](#) for the frozen expected-count version; [fit_mixed_subjects_mml\(\)](#) for the 2PL version.

Examples

```
set.seed(1)
pars <- data.frame(a = 1, d = c(-0.5, 0, 0.5))
observed <- simulate_2pl(rnorm(40), pars)
generated <- simulate_2pl(rnorm(100), pars)
fit <- fit_mixed_subjects_mml_1pl(
  observed, observed, generated,
  lambda = 0.5, initial_pars = pars, n_quad = 7,
  control = list(maxit = 100)
)
fit$item_pars
```

fit_mixed_subjects_split

Fit a split-sample mixed-subjects 2PL calibration

Description

Fits the same objective as [fit_mixed_subjects\(\)](#), but constructs labeled expected counts with cross-fitted posterior weights. For each split, the initial human 2PL model is fit on the other splits and then used to compute posterior weights for the held-out split. Each human row contributes to the final estimating equation exactly once.

Usage

```
fit_mixed_subjects_split(
  observed,
  predicted,
  generated,
  lambda = 1,
  n_splits = 2,
  split_id = NULL,
  seed = NULL,
  n_quad = 31,
  initial_pars = NULL,
  quadrature = NULL,
  common_predicted_weights = TRUE,
  paired_missing = c("match_observed", "allow"),
```

```

slope_lower = 1e-04,
slope_upper = NULL,
control = list(maxit = 500),
...
)

```

Arguments

observed	Human response matrix, with rows for subjects and columns for items. Values must be binary when <code>initial_pars</code> is omitted.
predicted	Binary LLM responses (0/1) for the same rows and items as observed. Probabilities are not accepted: fractional values are not a valid likelihood input for the marginal IRT objective and break the PPI correction, so sample binary responses from any probabilities first (e.g. <code>rbinom</code>).
generated	Binary generated or unlabeled LLM responses (0/1) for the same item columns. Probabilities are not accepted (see <code>predicted</code>).
lambda	Power-tuning parameter in $[0, 1]$. Supply a scalar for a fixed lambda or a vector with one value per split for a precomputed cross-fitted-lambda analysis. When a vector is supplied, the generated term uses the split-size weighted mean lambda.
n_splits	Number of sample splits.
split_id	Optional integer vector assigning each observed row to a split. If omitted, splits are sampled at random.
seed	Optional random seed used when <code>split_id</code> is omitted.
n_quad	Number of standard-normal quadrature nodes.
initial_pars	Optional item parameters to use in every fold instead of fitting fold-specific human models. This is mainly useful for testing or sensitivity analyses.
quadrature	Optional quadrature grid with <code>theta</code> and <code>weight</code> columns.
common_predicted_weights	Logical; if TRUE, reuse each held-out observed posterior weight matrix for its paired LLM responses.
paired_missing	How to handle missingness when <code>common_predicted_weights = TRUE</code> . The default, "match_observed", requires observed and predicted to have the same missingness pattern.
slope_lower	Lower bound for discrimination parameters during optimization. Use NULL for no lower bound.
slope_upper	Upper bound for discrimination parameters during optimization. Use NULL (the default) for no upper bound.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_2pl()</code> when fold-specific initial models are fit.

Details

Generated LLM counts are computed once per fold and averaged across folds so that the generated sample keeps its original sample-size scale.

Value

An object of class "mixedsubjects_fit" with split metadata and fold-level initial parameters.

Examples

```
set.seed(2)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
predicted <- observed
generated <- simulate_2pl(rnorm(100), pars)
fit <- fit_mixed_subjects_split(
  observed, predicted, generated,
  lambda = 0.5, initial_pars = pars, n_splits = 2,
  n_quad = 7, control = list(maxit = 50)
)
fit$item_pars
```

link_item_parameters *Link item parameters onto a target scale*

Description

Applies mean-mean linking to express source item parameters on the scale of a target calibration. Both parameter sets must be in slope-intercept form for the model $\text{plogis}(d + a * \text{theta})$.

Usage

```
link_item_parameters(source, target, method = c("mean_mean", "none"))
```

Arguments

source	Item parameters to transform. A matrix or data frame with columns a/a1 and d, or a fitted mirt model.
target	Item parameters defining the target scale. Uses the same accepted formats as source.
method	Linking method. Currently "mean_mean" and "none" are supported.

Details

If $\text{theta}_{\text{target}} = A * \text{theta}_{\text{source}} + B$, then source parameters transform as $a_{\text{target}} = a_{\text{source}} / A$ and $b_{\text{target}} = A * b_{\text{source}} + B$, with $d_{\text{target}} = -a_{\text{target}} * b_{\text{target}}$. Mean-mean linking chooses A and B so that the transformed source parameters match the target mean discrimination and mean difficulty.

Value

A list with transformed pars, linking constants A and B, and the selected method.

Examples

```
source <- data.frame(a = c(0.8, 1.2), d = c(-0.2, 0.5))
target <- data.frame(a = c(1.0, 1.5), d = c(-0.1, 0.4))
link_item_parameters(source, target)$pars
```

make_quadrature	<i>Create a standard-normal Gauss-Hermite quadrature grid</i>
-----------------	---

Description

`rmutil::gauss.hermite()` returns nodes and weights for integrals of the form $\int f(x) \exp(-x^2) dx$. This function rescales those nodes and weights to approximate expectations under a standard normal latent trait distribution.

Usage

```
make_quadrature(n_quad = 31, iterlim = 1e+05)
```

Arguments

<code>n_quad</code>	Number of quadrature nodes.
<code>iterlim</code>	Maximum number of Newton-Raphson iterations passed to <code>rmutil::gauss.hermite()</code> .

Value

A data frame with node index, theta, weight, and backward compatible aliases `X_k` and `A_k`.

Examples

```
quad <- make_quadrature(7)
sum(quad$weight)
```

mixed_subjects_loss	<i>Mixed-subjects objective function</i>
---------------------	--

Description

Evaluates the rectified mixed-subjects loss for 2PL item parameters. The parameter vector must contain all discriminations first, followed by all intercepts. The response probability is $\text{plogis}(d + a * \theta)$.

Usage

```
mixed_subjects_loss(pars, q_observed, q_predicted, q_11m, lambda = 0)
```

Arguments

<code>pars</code>	Numeric vector of item parameters: all discriminations a followed by all intercepts d .
<code>q_observed</code>	Quadrature summary for observed human responses, usually returned by <code>mixed_subjects_quadrature()</code> .
<code>q_predicted</code>	Quadrature summary for LLM responses/predictions on the same labeled human subjects.
<code>q_llm</code>	Quadrature summary for generated or unlabeled LLM responses.
<code>lambda</code>	Power-tuning parameter in $[0, 1]$.

Details

The objective is $L_{\text{observed}}(\text{pars}) + \lambda * (L_{\text{generated}}(\text{pars}) - L_{\text{predicted}}(\text{pars}))$. Setting $\lambda = 0$ gives the human-only expected-count objective.

Value

A scalar loss.

Examples

```
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- matrix(c(1, 0, 0, 1), nrow = 2, byrow = TRUE)
q <- mixed_subjects_quadrature(resp, item_pars = pars, N_quad = 5)
mixed_subjects_loss(c(pars$a, pars$d), q, q, q, lambda = 0.5)
```

`mixed_subjects_quadrature`

Convert responses to quadrature form

Description

Fits or accepts a 2PL model, computes posterior quadrature weights for each subject, and returns expected counts for mixed-subjects calibration. This is a lower-level helper; most analyses should call `fit_mixed_subjects()` or `fit_mixed_subjects_split()`.

Usage

```
mixed_subjects_quadrature(
  resp,
  N_quad = 31,
  eps = 1e-15,
  iterlim = 1e+05,
  irt_pars = NULL,
  item_pars = NULL,
  quadrature = NULL,
  link_method = "mean_mean",
  ...
)
```

Arguments

<code>resp</code>	A response matrix with rows for subjects and columns for items.
<code>N_quad</code>	Number of quadrature nodes to compute. Kept for backward compatibility; prefer <code>n_quad</code> in new code.
<code>eps</code>	Retained for backward compatibility. Stable log computations are used instead of probability clipping.
<code>iterlim</code>	Maximum number of Newton-Raphson iterations passed to <code>rmutil::gauss.hermite()</code> .
<code>irt_pars</code>	Optional target item parameters for mean-mean linking. This argument is kept for backward compatibility with earlier package versions.
<code>item_pars</code>	Optional item parameters. If omitted, a 2PL model is fit to <code>resp</code> using <code>fit_2pl()</code> .
<code>quadrature</code>	Optional quadrature grid with <code>theta</code> and <code>weight</code> columns.
<code>link_method</code>	Linking method used when <code>irt_pars</code> is supplied.
<code>...</code>	Additional arguments passed to <code>fit_2pl()</code> when <code>item_pars</code> is omitted.

Value

A list with `quad`, `counts`, `weights`, `irt_pars`, `quadrature`, and `theta`.

Examples

```
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- matrix(c(1, 0, 0, 1), nrow = 2, byrow = TRUE)
q <- mixed_subjects_quadrature(resp, item_pars = pars, N_quad = 5)
names(q)
```

`posterior_weights_2pl` *Compute posterior quadrature weights for a 2PL model*

Description

Computes each subject's posterior distribution over a fixed quadrature grid under a 2PL model, using stable log-likelihood calculations. Fractional responses in $[0, 1]$ are allowed at this low level, which is useful when LLM output is stored as probabilities rather than sampled binary responses.

Usage

```
posterior_weights_2pl(
  resp,
  item_pars,
  quadrature = NULL,
  n_quad = 31,
  iterlim = 1e+05
)
```

Arguments

resp	A response matrix with rows for subjects and columns for items. Values may be binary, fractional in $[0, 1]$, or NA.
item_pars	Item parameters in slope-intercept form. Supply a data frame or matrix with columns a/a1 and d, or a fitted mirt model.
quadrature	Optional quadrature data frame with theta and weight columns. If omitted, a standard-normal grid is created.
n_quad	Number of quadrature nodes used when quadrature is omitted.
iterlim	Maximum number of Newton-Raphson iterations passed to <code>rmutil::gauss.hermite()</code> when quadrature is omitted.

Details

Note: the high-level mixed-subjects fitting functions (`fit_mixed_subjects_mml()` and relatives) require **binary** predicted and generated; fractional input is supported only in these low-level quadrature utilities. If you have LLM-derived probabilities, sample binary responses from them (e.g. with `stats::rbinom()`) before calibrating.

Value

A matrix with one row per subject and one column per quadrature node. Rows sum to one. Attributes theta and weight contain the grid.

Examples

```

pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- matrix(c(1, 0, 0, 1), nrow = 2, byrow = TRUE)
W <- posterior_weights_2pl(resp, pars, n_quad = 5)
rowSums(W)

```

score_theta

Estimate ability scores from a 2PL calibration

Description

Computes bounded maximum-likelihood ability estimates for response patterns under fixed item parameters. This is a scoring helper for inspecting fitted calibrations; it does not account for uncertainty in the item parameters.

Usage

```
score_theta(resp, item_pars, bounds = c(-6, 6))
```

Arguments

resp	Response matrix with rows for subjects and columns for items.
item_pars	Item parameters in slope-intercept form. Supply a data frame or matrix with columns a/a1 and d, or a fitted mirt model.
bounds	Numeric vector of length two giving the optimization interval for theta.

Value

A numeric vector of ability estimates.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- simulate_2pl(rnorm(5), pars)
score_theta(resp, pars)
```

simulate_2pl

Simulate 2PL item responses

Description

Generates binary item responses from the model $\text{plogis}(d + a * \text{theta})$.

Usage

```
simulate_2pl(theta, item_pars)
```

Arguments

theta	Numeric vector of latent trait values.
item_pars	Item parameters in slope-intercept form. Supply a data frame or matrix with columns a/a1 and d, or a fitted mirt model.

Value

A binary response matrix with one row per value of theta and one column per item.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
simulate_2pl(rnorm(5), pars)
```

`summarize_expected_counts`*Summarize response data as expected quadrature counts*

Description

Converts response data and posterior quadrature weights into Bock-Aitkin style expected counts. For each item and quadrature node, N is the expected number of observed responses and R is the expected number correct.

Usage

```
summarize_expected_counts(resp, weights)
```

Arguments

`resp` A response matrix with rows for subjects and columns for items.
`weights` Posterior quadrature weights, usually returned by `posterior_weights_2pl()`.

Value

A list of class "mixedsubjects_counts" containing matrices N and R, sample size n, quadrature nodes, quadrature weights, and item names.

Examples

```
pars <- data.frame(a = c(1, 1.2), d = c(0, -0.5))
resp <- matrix(c(1, 0, 0, 1), nrow = 2, byrow = TRUE)
W <- posterior_weights_2pl(resp, pars, n_quad = 5)
counts <- summarize_expected_counts(resp, W)
counts$N
```

`tune_lambda_ability_risk`*Tune lambda by downstream ability-score risk*

Description

Fits candidate mixed-subjects calibrations, estimates the item-parameter sandwich covariance for each, and chooses the lambda that minimizes average propagated ability-score risk on a target response matrix.

Usage

```
tune_lambda_ability_risk(
  lambda_grid = seq(0, 1, by = 0.1),
  observed,
  predicted,
  generated,
  target_resp = NULL,
  theta_true = NULL,
  n_quad = 31,
  initial_pars = NULL,
  fit_fn = fit_mixed_subjects_mml,
  method = c("optimize", "grid"),
  bounds = c(-6, 6),
  max_discrimination = 10,
  control = list(maxit = 500),
  ...
)
```

Arguments

<code>lambda_grid</code>	Numeric vector of candidate lambda values in $[0, 1]$. For method = "grid" these are the evaluated candidates; for method = "optimize" only <code>range(lambda_grid)</code> matters and bounds the search (e.g. <code>lambda_grid = c(0, 0.8)</code> caps lambda at 0.8). Defaults to <code>seq(0, 1, by = 0.1)</code> .
<code>observed, predicted, generated</code>	Response matrices passed to <code>fit_mixed_subjects()</code> .
<code>target_resp</code>	Response matrix defining the target scoring population. If omitted, <code>observed</code> is used.
<code>theta_true</code>	Optional true theta values for <code>target_resp</code> , used in simulation studies to add squared scoring error to the risk. When omitted, <code>mean_squared_error</code> in the summary is NA; only <code>mean_param_var</code> is computed.
<code>n_quad</code>	Number of quadrature nodes.
<code>initial_pars</code>	Optional starting item parameters.
<code>fit_fn</code>	Fitting function to use. Defaults to <code>fit_mixed_subjects_mml()</code> , the marginal-likelihood PPI++ estimator (recommended). The frozen expected-count estimator <code>fit_mixed_subjects()</code> is still available by passing it here, but is discouraged : it has a gradient asymmetry that inflates discriminations and can drive lambda to 0 even for an informative predictor, and it requires a <code>slope_upper</code> cap for stability.
<code>method</code>	How lambda is chosen: "optimize" (default, direct 1-D optimization over <code>range(lambda_grid)</code> , continuous lambda) or "grid" (evaluate every value in <code>lambda_grid</code> and take the argmin).
<code>bounds</code>	Bounds passed to <code>score_theta()</code> .
<code>max_discrimination</code>	Upper bound on plausible item discrimination. Any candidate fit whose maximum $ a $ exceeds this value is treated as degenerate and excluded from selection. This guards against runaway discrimination fits, which can "converge"

with a spuriously low model-based risk (huge discrimination collapses the item-parameter covariance). The default of 10 is far above any realistic 2PL discrimination.

control Control list passed to `stats::optim()`.

... Additional arguments passed to `fit_fn`.

Details

This function minimizes $E[g' \Sigma_{\gamma} g]$ — the propagated ability-score risk — which is the appropriate objective for IRT applications where accurate test scoring is the goal. This is **distinct** from `tune_lambda_ppi_score()`, which minimizes the trace of the item-parameter covariance matrix $\text{Tr}(\Sigma_{\gamma})$ (the PPI++ theoretical objective). The two criteria generally yield different lambda values:

- `tune_lambda_ability_risk()` asks: which lambda produces the most accurate ability scores for the target population? Use this for operational scoring.
- `tune_lambda_ppi_score()` asks: which lambda minimizes item-parameter estimation variance? Use this for method validation and diagnostics.

Diagnostic note: if `tune_lambda_ability_risk()` selects $\lambda = 0$ for a misaligned LLM (one whose item parameters differ from the human calibration), this is the correct mathematical outcome under the current fixed-posterior expected-count implementation. The frozen posteriors create a gradient asymmetry that inflates item parameters at any $\lambda > 0$, increasing ability risk. This is not a bug in the risk function; it is a property of the estimating equations. See `fit_mixed_subjects_mml()` for a marginal-likelihood implementation that removes this asymmetry.

Tuning method. By default (`method = "optimize"`) lambda is selected by direct 1-D optimization (`stats::optimize()`) of the ability-score risk over the interval `range(lambda_grid)` (default `[0, 1]`), returning a *continuous* lambda with no grid rounding. With `method = "grid"` the risk is evaluated at each value of `lambda_grid` and the argmin returned (the previous behavior; useful for inspecting the whole risk surface). Both share the same runaway-discrimination guard and the same $\lambda = 0$ (human-only) fallback when no candidate is eligible.

Value

A list with `summary` (every evaluated lambda with its risk and diagnostics), `best_lambda` (continuous under `method = "optimize"`), `best_fit`, the evaluated fits and risks, and `method`.

See Also

`tune_lambda_ppi_score()` for the PPI++ theoretical lambda that minimizes the trace of the item-parameter covariance matrix; `fit_mixed_subjects_mml()` for the marginal-likelihood estimator.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
generated <- simulate_2pl(rnorm(100), pars)
tuned <- tune_lambda_ability_risk(
```

```

    c(0, 0.5), observed, observed, generated,
    initial_pars = pars, n_quad = 5, control = list(maxit = 30)
  )
  tuned$best_lambda

```

tune_lambda_ability_risk_1pl

Tune lambda by downstream ability-score risk for a 1PL model

Description

Selects the lambda minimizing $E[g' \text{Sigma}_{1pl} g]$ — the propagated ability-score risk in the 1PL parameterization — using `fit_mixed_subjects_mml_1pl()` by default. As in the 2PL `tune_lambda_ability_risk()`, lambda is chosen by direct 1-D optimization (method = "optimize", the default) or over `lambda_grid` (method = "grid").

Usage

```

tune_lambda_ability_risk_1pl(
  lambda_grid = seq(0, 1, by = 0.1),
  observed,
  predicted,
  generated,
  target_resp = NULL,
  theta_true = NULL,
  n_quad = 31,
  initial_pars = NULL,
  fit_fn = fit_mixed_subjects_mml_1pl,
  method = c("optimize", "grid"),
  bounds = c(-6, 6),
  max_discrimination = 10,
  control = list(maxit = 500),
  ...
)

```

Arguments

<code>lambda_grid</code>	Numeric vector of candidate lambda values in $[0, 1]$. For method = "grid" these are the evaluated candidates; for method = "optimize" only <code>range(lambda_grid)</code> matters and bounds the search (e.g. <code>lambda_grid = c(0, 0.8)</code> caps lambda at 0.8). Defaults to <code>seq(0, 1, by = 0.1)</code> .
<code>observed, predicted, generated</code>	Response matrices passed to <code>fit_mixed_subjects()</code> .
<code>target_resp</code>	Response matrix defining the target scoring population. If omitted, <code>observed</code> is used.

theta_true	Optional true theta values for target_resp, used in simulation studies to add squared scoring error to the risk. When omitted, mean_squared_error in the summary is NA; only mean_param_var is computed.
n_quad	Number of quadrature nodes.
initial_pars	Optional starting item parameters.
fit_fn	Fitting function. Defaults to <code>fit_mixed_subjects_mml_1pl()</code> .
method	How lambda is chosen: "optimize" (default, direct 1-D optimization over <code>range(lambda_grid)</code> , continuous lambda) or "grid" (evaluate every value in <code>lambda_grid</code> and take the argmin).
bounds	Bounds passed to <code>score_theta()</code> .
max_discrimination	Upper bound on plausible item discrimination. Any candidate fit whose maximum $ a $ exceeds this value is treated as degenerate and excluded from selection. This guards against runaway discrimination fits, which can "converge" with a spuriously low model-based risk (huge discrimination collapses the item-parameter covariance). The default of 10 is far above any realistic 2PL discrimination.
control	Control list passed to <code>stats::optim()</code> .
...	Additional arguments passed to <code>fit_fn</code> .

Details

Passes `fit_fn` to allow switching between the frozen expected-count estimator (`fit_mixed_subjects_1pl()`) and the marginal-MML estimator (`fit_mixed_subjects_mml_1pl()`).

Value

A list with `summary`, `best_lambda`, `best_fit`, `fits`, `risks`.

See Also

`tune_lambda_ability_risk()` for the 2PL version; `tune_lambda_ppi_score_1pl()` for the PPI++ score diagnostic.

Examples

```
set.seed(1)
pars <- data.frame(a = 1, d = c(-0.5, 0, 0.5))
obs <- simulate_2pl(rnorm(40), pars)
gen <- simulate_2pl(rnorm(100), pars)
tuned <- tune_lambda_ability_risk_1pl(
  c(0, 0.5), obs, obs, gen,
  initial_pars = pars, n_quad = 5, control = list(maxit = 30)
)
tuned$best_lambda
```

 tune_lambda_ability_risk_crossfit

Cross-fit ability-score-risk lambda tuning

Description

Estimates lambda separately for each held-out split using only the remaining labeled rows, then fits a final model. By default (`final_fit_fn = fit_mixed_subjects_mml`) the fold lambdas are averaged (weighted by fold size) into a single scalar and the full sample is refit; pass `final_fit_fn = fit_mixed_subjects_split` to instead fit each fold's rows with its own out-of-fold lambda.

Usage

```
tune_lambda_ability_risk_crossfit(
  lambda_grid = seq(0, 1, by = 0.1),
  observed,
  predicted,
  generated,
  target_resp = NULL,
  theta_true = NULL,
  n_splits = 2,
  split_id = NULL,
  seed = NULL,
  n_quad = 31,
  initial_pars = NULL,
  target_mode = c("fixed", "row_aligned"),
  fit_fn = fit_mixed_subjects_mml,
  final_fit_fn = fit_mixed_subjects_mml,
  tuning_args = list(),
  final_args = list(),
  bounds = c(-6, 6),
  control = list(maxit = 500),
  ...
)
```

Arguments

<code>lambda_grid</code>	Numeric vector of candidate lambda values in $[0, 1]$. For <code>method = "grid"</code> these are the evaluated candidates; for <code>method = "optimize"</code> only <code>range(lambda_grid)</code> matters and bounds the search (e.g. <code>lambda_grid = c(0, 0.8)</code> caps lambda at 0.8). Defaults to <code>seq(0, 1, by = 0.1)</code> .
<code>observed, predicted, generated</code>	Response matrices passed to <code>fit_mixed_subjects()</code> .
<code>target_resp</code>	Response matrix defining the target scoring population. If omitted, <code>observed</code> is used.

theta_true	Optional true theta values for target_resp, used in simulation studies to add squared scoring error to the risk. When omitted, mean_squared_error in the summary is NA; only mean_param_var is computed.
n_splits	Number of sample splits.
split_id	Optional integer split assignment for labeled rows.
seed	Optional seed used when split_id is omitted.
n_quad	Number of quadrature nodes.
initial_pars	Optional starting item parameters.
target_mode	How target_resp is handled in each fold. "fixed" (default): the full target_resp is used to evaluate risk in every fold, suitable when the target population is fixed and independent of the labeled-data split (e.g. an operational scoring population). "row_aligned": only the training rows of target_resp are used, which is valid when target_resp = observed and fold-matched evaluation is desired.
fit_fn	Fitting function used for each fold's ability-risk tuning (passed to <code>tune_lambda_ability_risk()</code>). Defaults to <code>fit_mixed_subjects_mml()</code> (marginal MML, recommended). The frozen expected-count estimator <code>fit_mixed_subjects()</code> is still available but discouraged.
final_fit_fn	Function used to produce the final combined-data fit. Defaults to <code>fit_mixed_subjects_mml()</code> , giving a scalar marginal-MML final fit: the fold-specific lambdas are averaged (weighted by fold size) into a single scalar and the full sample is refit. Pass <code>fit_mixed_subjects_split()</code> to instead keep the per-fold lambda vector and fit each fold's rows with its own out-of-fold lambda — the textbook cross-fit decoupling, but it uses the discouraged frozen expected-count split estimator.
tuning_args	Named list of extra arguments forwarded only to the fold-level <code>tune_lambda_ability_risk()</code> calls (and through them to <code>fit_fn</code>). For example, <code>tuning_args = list(slope_upper = 4)</code> .
final_args	Named list of extra arguments forwarded only to <code>final_fit_fn</code> . For example, <code>final_args = list(mml_pred_weights = "own")</code> . This keeps tuning-specific and final-fit-specific arguments cleanly separated, avoiding the earlier ... leakage between the two.
bounds	Bounds passed to <code>score_theta()</code> .
control	Control list passed to <code>stats::optim()</code> .
...	Deprecated; forwarded to <code>tuning_args</code> for backward compatibility with a one-time message. Prefer <code>tuning_args</code> / <code>final_args</code> .

Value

A list with fold-specific lambda values, fold tuning objects, and the final fit.

tune_lambda_ability_risk_item

Per-item ability-risk lambda tuning via coordinate descent

Description

Finds a per-item vector of lambda values λ_j in $[0, 1]$ that minimizes propagated ability-score risk $E[g' \Sigma_{\gamma} g]$ using coordinate descent on the items. Each coordinate step holds the other $\lambda_{\{j\}}$ fixed and selects λ_j by direct 1-D optimization (method = "optimize", the default, continuous) or over λ_{grid} (method = "grid").

Usage

```
tune_lambda_ability_risk_item(
  lambda_grid = seq(0, 1, by = 0.1),
  observed,
  predicted,
  generated,
  target_resp = NULL,
  theta_true = NULL,
  n_quad = 31,
  initial_pars = NULL,
  n_pass = 1,
  init_lambda = 0,
  method = c("optimize", "grid"),
  bounds = c(-6, 6),
  max_discrimination = 10,
  control = list(maxit = 300),
  ...
)
```

Arguments

lambda_grid	Numeric vector of candidate lambda values in $[0, 1]$ to try for each item independently.
observed, predicted, generated	Response matrices passed to <code>fit_mixed_subjects_mml()</code> .
target_resp	Target scoring population. If omitted, observed is used.
theta_true	Optional true theta values, used to add squared scoring error to the risk.
n_quad	Number of quadrature nodes.
initial_pars	Optional starting item parameters.
n_pass	Number of coordinate-descent passes (default 1).
init_lambda	Starting lambda vector for coordinate descent. Supply the global scalar optimum from <code>tune_lambda_ability_risk()</code> (e.g. <code>init_lambda = 0.5</code>) to start

	the search around a useful operating point. Starting from all-zeros is not recommended: each single-item improvement is too small to detect when other items are at zero. A scalar is broadcast to all items; a vector of length <code>n_items</code> sets per-item starting values.
<code>method</code>	How each item's lambda is chosen at a coordinate step: "optimize" (default, direct 1-D optimization over <code>range(lambda_grid)</code> , continuous) or "grid" (evaluate every value in <code>lambda_grid</code>).
<code>bounds</code>	Bounds passed to <code>score_theta()</code> .
<code>max_discrimination</code>	Upper bound on plausible item discrimination; any candidate fit whose maximum $ a $ exceeds it is treated as degenerate and skipped. See <code>tune_lambda_ability_risk()</code> for the rationale. Default 10.
<code>control</code>	Control list passed to <code>stats::optim()</code> .
<code>...</code>	Additional arguments passed to <code>fit_mixed_subjects_mml()</code> .

Details

Calls `fit_mixed_subjects_mml()` with a per-item lambda vector at each candidate evaluation. Because the lambda is a vector, that function **switches to its frozen expected-count Q-function path** — posteriors are frozen at `initial_pars`, not recomputed continuously. This is an approximation; see the @note below. The resulting lambda vector can be used directly with `fit_mixed_subjects_mml()`.

Computational cost. Each pass refits per item per candidate lambda: `method = "grid"` does `n_items × length(lambda_grid)` fits; `method = "optimize"` does roughly `n_items × 12` (the optimizer's evaluations plus the endpoints). Use `n_pass = 1` (the default) for a single greedy sweep, which is usually sufficient.

Value

A list with `lambda` (per-item vector), `item` (item names), `n_pass`, `method`, and `final_fit` (the `fit_mixed_subjects_mml()` fit at the selected lambda).

Note

Approximation status. The coordinate descent fits use the frozen expected-count Q-function (not the full marginal-MML objective) because the IRT marginal likelihood integrates over the joint response pattern and does not decompose item-wise. The approach is approximately correct when `initial_pars` is close to the converged parameters. Report per-item results as experimental / approximate.

See Also

`tune_lambda_ppi_score_item()` for the faster PPI++-score version; `tune_lambda_ability_risk()` for the global scalar version.

Examples

```

set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
generated <- simulate_2pl(rnorm(100), pars)
tuned <- tune_lambda_ability_risk_item(
  c(0, 0.5), observed, observed, generated,
  initial_pars = pars, n_quad = 5, control = list(maxit = 30)
)
tuned$lambda

```

tune_lambda_ppi_score *Plug-in PPI++ optimal tuning parameter*

Description

Implements the closed-form estimator from Proposition 2 of Angelopoulos, Duchi and Zrnic (2023) for the lambda that minimizes the trace of the asymptotic item-parameter covariance matrix $\text{Tr}(\text{Sigma}_{\text{gamma}})$.

Usage

```

tune_lambda_ppi_score(
  observed,
  predicted,
  item_pars,
  n_generated,
  quadrature = NULL,
  n_quad = 31
)

```

Arguments

observed	Human response matrix.
predicted	Paired binary LLM responses (0/1) for the same rows as observed. Probabilities are not accepted; sample binary responses first.
item_pars	Item parameters in slope-intercept form at which to evaluate the score vectors. Typically the human 2PL MLE from <code>fit_2pl()</code> .
n_generated	Number of generated (unpaired) LLM subjects, used to compute the ratio r ($n / n_{\text{generated}}$).
quadrature	Optional quadrature grid. If omitted, a standard-normal grid with <code>n_quad</code> nodes is created.
n_quad	Number of quadrature nodes when quadrature is omitted.

Details

This is the item-parameter variance objective, not the psychometric scoring objective. For IRT applications where accurate ability scoring is the goal, use `tune_lambda_ability_risk()` or `tune_lambda_ability_risk_crossfit()` instead. Those functions directly minimize the propagated ability-score risk $E[g' \Sigma_{\gamma} g]$ — the quantity that matters for test scoring — rather than item-parameter estimation efficiency. `tune_lambda_ppi_score()` is provided as a theoretical diagnostic and to facilitate method validation.

The formula uses the **same** human posterior weights for both the human and paired-LLM score vectors. This symmetry is required for the PPI++ unbiasedness condition $E[\text{grad}_{\text{gen}}] = E[\text{grad}_{\text{pred}}]$ at the true parameters.

Value

A list with elements `lambda` (the plug-in estimate, clipped to $[0, 1]$), `n`, `n_generated`, `r`, and the intermediate matrices `C_hf` (cross-covariance of human and paired-LLM score vectors) and `V_f` (variance of paired-LLM score vectors).

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
predicted <- observed
tune_lambda_ppi_score(observed, predicted, pars, n_generated = 100, n_quad = 7)$lambda
```

tune_lambda_ppi_score_1pl

Plug-in PPI++ optimal tuning parameter for a IPL model

Description

Applies the PPI++ Proposition 2 formula using $(J+1)$ -dimensional score vectors for the IPL parameterization $(a_{\text{shared}}, d_1, \dots, d_J)$.

Usage

```
tune_lambda_ppi_score_1pl(
  observed,
  predicted,
  item_pars,
  n_generated,
  quadrature = NULL,
  n_quad = 31
)
```

Arguments

observed	Human response matrix.
predicted	Paired binary LLM responses (0/1) for the same rows as observed. Probabilities are not accepted; sample binary responses first.
item_pars	Item parameters in slope-intercept form at which to evaluate the score vectors. Typically the human 2PL MLE from <code>fit_2pl()</code> .
n_generated	Number of generated (unpaired) LLM subjects, used to compute the ratio r ($n / n_generated$).
quadrature	Optional quadrature grid. If omitted, a standard-normal grid with n_quad nodes is created.
n_quad	Number of quadrature nodes when quadrature is omitted.

Details

This is the **item-parameter variance** objective — it minimizes $\text{Tr}(\text{Sigma}_{1pl})$. For practical scoring applications use `tune_lambda_ability_risk_1pl()` instead.

Value

A list with `lambda`, `n`, `n_generated`, `r`, `C_hf`, `V_f`.

Examples

```
set.seed(1)
pars <- data.frame(a = 1, d = c(-0.5, 0, 0.5))
obs <- simulate_2pl(rnorm(40), pars)
tune_lambda_ppi_score_1pl(obs, obs, pars, n_generated = 100, n_quad = 7)$lambda
```

tune_lambda_ppi_score_item

Per-item PPI++ optimal tuning parameters

Description

Applies the PPI++ Proposition 2 plug-in formula independently for each item, producing a vector of item-specific lambda values `lambda_j` in $[0, 1]$.

Usage

```
tune_lambda_ppi_score_item(
  observed,
  predicted,
  item_pars,
  n_generated,
  quadrature = NULL,
  n_quad = 31
)
```

Arguments

observed	Human response matrix.
predicted	Paired binary LLM responses (0/1) for the same rows as observed. Probabilities are not accepted; sample binary responses first.
item_pars	Item parameters at which to evaluate the score vectors.
n_generated	Number of generated (unpaired) LLM subjects.
quadrature	Optional quadrature grid.
n_quad	Number of quadrature nodes when quadrature is omitted.

Details

The global `tune_lambda_ppi_score()` uses the full parameter covariance matrix $\text{Tr}(\text{Sigma_gamma})$ as the objective. This function instead applies the same formula using only the 2x2 diagonal block of the inverse Hessian for item j , and the 2D sub-vectors of the human and paired-LLM score vectors. The result is the lambda that minimizes the marginal variance of (a_j, d_j) independently for each item.

Use case. When a single global lambda is forced to zero because a few items have poor LLM predictions, per-item lambda_j allows well-predicted items to still benefit from the LLM data. Pass the returned vector to `fit_mixed_subjects_mml()` as the lambda argument.

This is a **theoretical diagnostic**: it minimizes item-parameter variance, not ability-score risk. For operational scoring use `tune_lambda_ability_risk_item()` instead.

Value

A list with lambda (numeric vector of length n_items), item (item names), n_generated, and r (the ratio n / n_generated).

See Also

`tune_lambda_ppi_score()` for the global version; `fit_mixed_subjects_mml()` to fit with a per-item lambda vector.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
tune_lambda_ppi_score_item(observed, observed, pars, n_generated = 100, n_quad = 7)$lambda
```

vcov_mixed_subjects *Sandwich covariance for a mixed-subjects fit*

Description

Estimates the full sandwich covariance matrix for item parameters from the fixed-posterior expected-count estimating equations. The parameter order is all discriminations followed by all intercepts, matching `fit$par`.

Usage

```
vcov_mixed_subjects(object, ridge = 1e-08, ...)
```

Arguments

object	A fitted object returned by <code>fit_mixed_subjects()</code> or <code>fit_mixed_subjects_from_quadrature()</code> with response matrices and posterior weights available in its quadrature summaries.
ridge	Small ridge value used when inverting the Hessian.
...	Unused; included for method compatibility.

Value

A covariance matrix with attributes `bread` and `meat`.

Examples

```
set.seed(1)
pars <- data.frame(a = c(1, 1.2, 0.9), d = c(0, -0.5, 0.3))
observed <- simulate_2pl(rnorm(40), pars)
fit <- fit_mixed_subjects(
  observed, observed, simulate_2pl(rnorm(80), pars),
  lambda = 0.5, initial_pars = pars, n_quad = 7
)
dim(vcov_mixed_subjects(fit))
```

vcov_mixed_subjects_1pl

Sandwich covariance for a 1PL mixed-subjects fit

Description

Estimates the $(J+1) \times (J+1)$ sandwich covariance matrix for the shared discrimination and per-item intercepts of a 1PL mixed-subjects calibration.

Usage

```
vcov_mixed_subjects_1pl(object, ridge = 1e-08, ...)
```

Arguments

object	A "mixedsubjects_1pl_fit" object from fit_mixed_subjects_1pl() or fit_mixed_subjects_mml() .
ridge	Ridge regularization for Hessian inversion.
...	Unused.

Value

A $(J+1) \times (J+1)$ covariance matrix. Row/column names are "a_shared" and "d_Item1", "d_Item2", etc.

Note

Bread approximation. The bread uses `avg_hessian_counts_1pl()`, the EM complete-data Hessian for the 1PL model, rather than the Louis (1982) marginal observed-information correction implemented for 2PL in `vcov_mixed_subjects_mml()`. The EM bread over-states efficiency by ignoring missing information about theta. A Louis-corrected 1PL bread is planned for a future release.

```
vcov_mixed_subjects_mml
```

Marginal-MML sandwich covariance for a mixed-subjects fit

Description

Computes the full sandwich covariance for the scalar marginal-MML PPI++ estimator from [fit_mixed_subjects_mml\(\)](#). The bread uses Louis's (1982) observed marginal-information formula

Usage

```
vcov_mixed_subjects_mml(object, ridge = 1e-08, ...)
```

Arguments

object	A scalar-lambda fit_mixed_subjects_mml() fit.
ridge	Ridge regularization for bread inversion.
...	Unused.

Details

$$A_{\lambda}^{\text{marg}} = H_{\lambda}^{\text{comp}} - I_{\lambda}^{\text{miss}}$$

rather than the EM/complete-data Hessian used by `vcov_mixed_subjects()`. Using the complete-data Hessian as the bread for a marginal-MML estimator would over-state efficiency by ignoring the missing-information correction.

The meat uses the standard marginal per-person score vectors (posteriors at the converged parameters), which is identical to `vcov_mixed_subjects()`.

When is this function called automatically? The `vcov()` method for "mixedsubjects_fit" objects (see `stats::vcov()`) dispatches here whenever `isTRUE(object$mml) && length(object$lambda) == 1`. For vector-lambda fits, or for frozen expected-count fits, the existing `vcov_mixed_subjects()` is used.

Value

A $2J \times 2J$ covariance matrix with attributes `bread` and `meat`.

See Also

`vcov_mixed_subjects()` for the frozen expected-count version. The internal `louis_missing_info()` helper computes the missing-information correction.

Index

ability_gradient, 3
ability_gradient_1pl, 3
ability_risk, 4
ability_risk(), 5
ability_risk_1pl, 5

diagnose_lambda_grid, 6

fit_1pl, 7
fit_1pl(), 11, 18
fit_2pl, 8
fit_2pl(), 10, 14, 16, 20, 24, 36, 38
fit_mixed_subjects, 9
fit_mixed_subjects(), 6, 10, 12, 13, 15–19,
23, 28, 30, 32, 33, 40
fit_mixed_subjects_1pl, 10
fit_mixed_subjects_1pl(), 19, 31, 41
fit_mixed_subjects_from_quadrature, 12
fit_mixed_subjects_from_quadrature(),
40
fit_mixed_subjects_iterative, 13
fit_mixed_subjects_mml, 15
fit_mixed_subjects_mml(), 17, 19, 25, 28,
29, 33–35, 39, 41
fit_mixed_subjects_mml_1pl, 17
fit_mixed_subjects_mml_1pl(), 12, 30, 31,
41
fit_mixed_subjects_split, 19
fit_mixed_subjects_split(), 6, 23, 33

link_item_parameters, 21

make_quadrature, 22
mirt::mirt(), 8
mixed_subjects_loss, 22
mixed_subjects_quadrature, 23
mixed_subjects_quadrature(), 12, 23

posterior_weights_2pl, 24
posterior_weights_2pl(), 27

rmutil::gauss.hermite(), 22, 24, 25

score_theta, 25
score_theta(), 3–5, 28, 31, 33, 35
simulate_2pl, 26
stats::optim(), 7, 10, 11, 13, 14, 16, 18, 20,
29, 31, 33, 35
stats::optimize(), 29
stats::rbinom(), 25
stats::vcov(), 42
summarize_expected_counts, 27
summarize_expected_counts(), 12

tune_lambda_ability_risk, 27
tune_lambda_ability_risk(), 14–16, 30,
31, 33–35, 37
tune_lambda_ability_risk_1pl, 30
tune_lambda_ability_risk_1pl(), 38
tune_lambda_ability_risk_crossfit, 32
tune_lambda_ability_risk_crossfit(),
37
tune_lambda_ability_risk_item, 34
tune_lambda_ability_risk_item(), 17, 39
tune_lambda_ppi_score, 36
tune_lambda_ppi_score(), 15, 29, 39
tune_lambda_ppi_score_1pl, 37
tune_lambda_ppi_score_1pl(), 31
tune_lambda_ppi_score_item, 38
tune_lambda_ppi_score_item(), 35

vcov_mixed_subjects, 40
vcov_mixed_subjects(), 17, 42
vcov_mixed_subjects_1pl, 40
vcov_mixed_subjects_1pl(), 5
vcov_mixed_subjects_mml, 41
vcov_mixed_subjects_mml(), 16, 17, 41