Package 'bayesqm'

August 20, 2026

Type Package

Title Bayesian Q Methodology: Exact Rank-Order Likelihood for Forced Q Sorts

Version 0.2.0

Date 2026-08-19

Description A Bayesian analysis for Q methodology, alongside the classical one. Models the forced Q sort as an ordered partition of the statements through an exact rank-order likelihood (the design quotas fix the partition margins, so the likelihood of the observed sorting event is exact), fits it by a parameter-expanded Gibbs sampler in R with no compiled code and a convergence gate on rotation-invariant functionals, resolves rotational ambiguity via the MatchAlign post-processing of Poworoznek et al. (2025) <doi:10.1214/25-BA1544>, and returns the familiar Q tables as posterior summaries: credible intervals for bounded participant loadings, flag probabilities with an explicit unclassified state, quota-respecting factor arrays, distinguishing and consensus statements judged against a posterior critical difference and a grid-width equivalence region, one posterior false-discovery rule for all published claims, and a two-signal posterior-predictive workflow for the number of factors.

License GPL (>= 3)

URL https://github.com/rdazadda/bayesqm, https://rdazadda.github.io/bayesqm/

BugReports https://github.com/rdazadda/bayesqm/issues

Encoding UTF-8

Language en-US

LazyData true

Config/testthat/edition 3

Depends R (>= 4.1.0)

Imports stats, utils, tools, clue, posterior (>= 1.5.0)

Suggests loo (>= 2.7.0), readxl (>= 1.4.0), jsonlite (>= 1.8.0), ggplot2 (>= 3.4.0), knitr (>= 1.40), rmarkdown (>= 2.20), testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website qmethod

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Raymond Dacosta Azadda [aut, cre] (ORCID: <https://orcid.org/0009-0002-4384-4556>), Henry Ofoe Agbi-Kaiser [aut] (ORCID: <https://orcid.org/0009-0008-6127-9136>), Hannah D. Robinson [aut] (ORCID: <https://orcid.org/0009-0000-5159-1363>), AK-ACE Team [aut], Karsten Hueffer [aut], Taa'aii Peter [aut], Stacy Rasmus [aut]

Maintainer Raymond Dacosta Azadda <rdazadda@alaska.edu>

Repository CRAN

Date/Publication 2026-08-20 10:52:15 UTC

Contents

`assess_recovery` *Simulation-study assessment helpers*

Description

`assess_recovery()` compares a point estimate and optional posterior draws to a known loading truth, returning RMSE, per-factor RMSE, bias, Tucker's congruence, credible-interval coverage, width, and Gneiting-Raftery interval score. `assess_classification()` compares a logical flag matrix to the true factor assignments using optimal permutation matching.

Usage

```
assess_recovery(Lambda_hat, Lambda_true, Lambda_draws = NULL, prob = 0.95)

assess_classification(flags, Lambda_true)
```

Arguments

Lambda_hat	Estimated loading matrix (N x K).
Lambda_true	True loading matrix.
Lambda_draws	Optional array of shape [T, N, K] of posterior draws, used for coverage / interval metrics.
prob	Credible-interval probability.
flags	Logical matrix of factor assignments.

Value

assess_recovery() returns a list of metrics; assess_classification() returns a list with accuracy and per_factor.

---

bayesqm-colors *Get or set the bayesqm colour scheme*

Description

Every plot in the package reads its palette through bayesqm_colors(). Call bayesqm_set_colors() to switch the active scheme for every subsequent plot. The available built-in schemes are "sorts" (default, the plot_sorts() colour family), "blue", "teal", "red", "purple", and "grey". For full control, pass a named list with slots dark, accent, grey, gridgrey, and fill.

Usage

```
bayesqm_colors()

bayesqm_set_colors(scheme)
```

Arguments

scheme	Character name of a built-in scheme, or a named list of colours with the slot names listed in the description.

Value

bayesqm_colors() returns the active palette as a named list. bayesqm_set_colors() returns the previous scheme name, invisibly.

Examples

```
bayesqm_colors()
old <- bayesqm_set_colors("teal")
bayesqm_colors()[["fill"]]
bayesqm_set_colors(old)
```

caption_bayesqm *Caption text for figures from a fit*

Description

A single string naming the model, the panel, the draws, and the gate verdict — the provenance a figure caption should carry.

Usage

```
caption_bayesqm(fit, include_gate = TRUE)
```

Arguments

fit	A bayesqm_fit.
include_gate	Append the gate report (default TRUE).

Value

A length-one character string.

Examples

```
caption_bayesqm(demo_fit())
```

check_fit *Posterior-predictive checks of the fitted model*

Description

Three checks, each comparing the observed sorts with replicates drawn from the fitted model: the agreement check (T1a) compares the observed person-agreement matrix with replicated ones against a double-replicate reference; the extra-factor check (T1b) locates the observed (K+1)-th eigenvalue in its replicated distribution; and the paired-comparison check (T2) compares standardized statement-pair margins, ties counted one half. These are diagnostics to read, not tests to pass.

Usage

```
check_fit(fit, draws = 100)
```

Arguments

fit	A bayesqm_fit.
draws	Posterior draws used for replication (default 100).

Value

A bayesqm_checks list: agreement (p), extra_factor (percentile), and paired (p, per-statement p_j), with a print method.

Examples

```
check_fit(demo_fit(), draws = 20)
```

---

check_persons *The person check against mixed-replication bands*

Description

For each participant, the model-agreement statistic m (mean over draws of the best absolute Spearman agreement with any factor array) and the person-agreement statistic w (best absolute agreement with any other sort), each located against bands from mixed replicates (fresh persons drawn from the fitted model). Verdicts: fits, no_shared (below the model band, inside the person band), unspanned (below the model band but above the person band, a shared viewpoint the fitted factors do not span), and atypical. Unspanned persons name their nearest partner.

Usage

```
check_persons(fit, draws = 60, mixes = 150)
```

Arguments

fit	A bayesqm_fit.
draws	Draw grid for the arrays (default 60).
mixes	Mixed replicates for the bands (default 150).

Value

A bayesqm_persons data frame: participant, m, w, partner, verdict, with the band limits attached as attributes.

Examples

```
check_persons(demo_fit(), draws = 20, mixes = 40)
```

claims *Selected claims at a common false-discovery level*

Description

Selects every claim — participant flags, distinguishing listings, consensus statements, and pairwise stars — by the one posterior false-discovery rule at level q, and reports the expected number of false claims per family alongside. The same code path produces the `selected` columns of `compute_flags()` and `compute_qdc()`, so the tables and this object always agree.

Usage

```
claims(fit, q = 0.05, flag_floor = 0.5, cons_floor = 0.95)
```

Arguments

`fit`	A `bayesqm_fit`.
`q`	Posterior expected false-discovery bound (default 0.05).
`flag_floor, cons_floor`	Selection floors for flags (0.5) and consensus statements (0.95).

Value

A `bayesqm_claims` object: selected-only tables `flags`, `distinguishing`, `consensus`, and `stars`, each with its `expected_false` count, plus the level `q`.

Examples

```
claims(demo_fit())
```

coef.bayesqm_fit *Posterior-mean bounded loadings*

Description

Posterior-mean bounded loadings

Usage

```
## S3 method for class 'bayesqm_fit'
coef(object, ...)
```

Arguments

object	A bayesqm_fit.
...	Unused.

Value

An N x K matrix of posterior-mean bounded loadings (the correlation-scale loading of the accompanying paper).

---

compute_factor_array *Quota-respecting factor arrays*

Description

The reported array for each factor: statements sorted onto the design grid by their posterior column probabilities (mean grid column, tie-broken by posterior-mean score), which is quota-exact by construction. When the **clue** package is installed, a footrule assignment cross-check is computed and its disagreement rate attached.

Usage

```
compute_factor_array(fit, negative = FALSE)
```

Arguments

fit	A bayesqm_fit.
negative	Also return the negative-pole arrays (default FALSE); on a symmetric grid the mirror is exact.

Value

A data frame with one row per statement: statement, then f{k}_grid per factor (and f{k}_grid_neg when negative = TRUE). attr(, "footrule_disagreement") gives the share of cells where the footrule assignment differs (NA without **clue**).

Examples

```
compute_factor_array(demo_fit())
```

compute_flags *Flag probabilities with an explicit unclassified state*

Description

For each participant, the posterior probability that the classical flag rule fires on each signed factor, with the probability of remaining unclassified alongside. Selected flags come from all signed candidates by the posterior false-discovery rule at level q.

Usage

```
compute_flags(fit, q = 0.05, floor = 0.5)
```

Arguments

fit	A bayesqm_fit.
q	Posterior expected false-discovery bound (default 0.05).
floor	Minimum flag probability a candidate needs before it can be selected (default 0.5, so at most one signed flag per participant).

Value

A data frame with one row per participant: modal signed candidate (factor, sign), its probability flag_prob, unclassified_prob, and selected. The full probability matrix is attached as attr(, "phi") and the expected number of false selected flags as attr(, "expected_false").

Examples

```
compute_flags(demo_fit())
```

compute_loadings *Bounded participant loadings with credible intervals*

Description

The correlation-scale loading of the accompanying paper, rho = (S lambda)_k / sqrt(s^2 + 1), summarized per participant and factor, with the posterior-mean person spread s_i alongside.

Usage

```
compute_loadings(fit, prob = NULL)
```

Arguments

fit	A bayesqm_fit.
prob	Credible-interval probability; defaults to the fit's.

Value

A data frame with one row per participant: participant, then f{k}_loading, f{k}_lower, f{k}_upper per factor, then spread (posterior-mean s_i).

Examples

```
compute_loadings(demo_fit())
```

---

compute_qdc *Distinguishing and consensus verdicts*

Description

The statement verdict table. Distinguishing-for-a-factor is the joint event that every contrast touching that factor exceeds the posterior critical difference delta_kl (computed from the posterior spread of the score contrasts); consensus is the equivalence event that all scores sit within one grid column (delta_grid()) of each other. Both families are selected through the posterior false-discovery rule, and a statement can be distinguishing, consensus, or indeterminate.

Usage

```
compute_qdc(fit, q = 0.05, cons_floor = 0.95, prob = NULL)
```

Arguments

fit	A bayesqm_fit with K >= 2.
q	Posterior expected false-discovery bound (default 0.05).
cons_floor	Minimum consensus probability before a statement can be selected as consensus (default 0.95).
prob	Credible-interval probability for the contrast intervals; defaults to the fit's. The critical difference itself keeps its z_0.975 definition regardless.

Value

A data frame with one row per statement: per-factor distinguishing probabilities, consensus_prob, and the verdict. Attributes: delta_kl and delta_kl99 (per pair), delta_grid, contrasts (the per-pair contrast table with intervals, the probability diff_column_prob that the two viewpoints place the statement in different grid columns, and stars, * for selected claims and ** for those that also clear the z_0.995 critical difference at probability .99), and expected_false per family.

Examples

```
compute_qdc(demo_fit())
```

`compute_zscores` *Statement scores with credible intervals*

Description

Statement scores with credible intervals

Usage

```
compute_zscores(fit, prob = NULL)
```

Arguments

`fit`	A `bayesqm_fit`.
`prob`	Credible-interval probability; defaults to the fit's.

Value

A data frame with one row per statement: `statement`, then `f{k}_zsc`, `f{k}_lower`, `f{k}_upper` per factor.

Examples

```
compute_zscores(demo_fit())
```

`crib_sheet` *Extreme-placement probabilities per statement and factor*

Description

The crib sheet: for each statement and factor, the posterior probability of landing in the top or bottom grid column of that factor's array, and of carrying the highest or lowest score across factors.

Usage

```
crib_sheet(fit)
```

Arguments

`fit`	A `bayesqm_fit`.

Value

A long data frame: `statement`, `factor`, `p_top`, `p_bottom`, `p_highest`, `p_lowest`.

Examples

```
head(crib_sheet(demo_fit()))
```

`delta_grid` *Grid width on the z scale*

Description

The width of one grid column on the standardized score scale, `1 / sd(forced positions)` with the population standard deviation. This is the equivalence-region half-width used for consensus verdicts: a score contrast smaller than one grid column is not expressible in a completed sort.

Usage

```
delta_grid(distribution)
```

Arguments

`distribution`	Integer vector of forced-distribution counts (the number of statements allowed in each grid column).

Value

A single numeric value.

Examples

```
delta_grid(c(2, 3, 4, 5, 7, 7, 5, 4, 3, 2))
```

demo_fit *A small demonstration fit*

Description

Runs the partition-model sampler on a small synthetic forced-sort dataset with a fixed seed and the convergence gate disabled. It exists so examples and tests have a complete `bayesqm_fit` in about a second; it is no substitute for `fit_bayesian()` at its defaults on real data.

Usage

```
demo_fit(N = 8, J = 13, K = 2, draws = 200, seed = 1)
```

Arguments

`N`, `J`, `K`	Panel size, statement count, factors (defaults 8, 13, 2).
`draws`	Kept posterior draws (default 200).
`seed`	Random seed (default 1).

Value

A `bayesqm_fit`.

Examples

```
fit <- demo_fit()
fit
```

extend *Continue sampling a fitted chain*

Description

Warm-extends the fitted chain by further iterations and re-runs the convergence gate and the alignment on the grown draw set. The extension restores the sampler's saved random-number state, so the result is draw-for-draw identical to having run one longer chain, whatever happened in the session in between.

Usage

```
extend(fit, iterations = NULL, pivot = NULL, quiet = FALSE)
```

Arguments

fit	A bayesqm_fit created with keep_raw = TRUE.
iterations	Additional iterations; NULL (the default) doubles the chain. Must be a multiple of the fit's thin.
pivot, quiet	As in fit_bayesian().

Value

The extended bayesqm_fit.

---

factor_characteristics

Factor characteristics

Description

The per-factor block of the accompanying paper: modal and mean number of defining sorts per posterior draw with a credible interval, the selected flag count, the mean posterior spread of the statement scores, and the mean replicate reliability R_i = s_i^2 / (1 + s_i^2) of the flagged participants. The statement-score correlations between factors ride along as an attribute.

Usage

```
factor_characteristics(fit, prob = 0.9, q = 0.05, floor = 0.5)
```

Arguments

fit	A bayesqm_fit.
prob	Credible-interval probability for the defining-sort counts (default 0.90).
q, floor	The flag rule fixing the flagged set, as in compute_flags().

Value

A data frame with one row per factor: factor, flagged (selected flags), defining_modal, defining_mean, defining_lower, defining_upper, score_spread, and reliability. Attribute score_correlations holds the posterior mean K x K correlation matrix of the statement scores.

Examples

```
factor_characteristics(demo_fit())
```

fitted.bayesqm_fit *Posterior-mean reconstruction on the utility scale*

Description

Posterior-mean reconstruction on the utility scale

Usage

```
## S3 method for class 'bayesqm_fit'
fitted(object, ...)
```

Arguments

object	A bayesqm_fit.
...	Unused.

Value

A J x N matrix of posterior-mean latent utilities F Lambda'.

fit_bayesian *Fit the exact partition-likelihood model to forced Q sorts*

Description

Models each completed sort as a forced ordered partition of the statements and samples the posterior of the latent factor model by a parameter-expanded Gibbs sampler. Convergence is gated on rotation-invariant person spreads (rank-normalized split-R-hat and bulk/tail effective sample size); a chain failing the gate is warm-extended at successive doublings up to max_iterations. Draws are aligned by MatchAlign with a polarity canon, so defining sorts load positively.

Usage

```
fit_bayesian(
  Y,
  K,
  iterations = 12000,
  burn = 2000,
  thin = 5,
  max_iterations = 48000,
  seed = NULL,
  sigma_scale = 1,
  rhat_max = 1.01,
  ess_min = 400,
```

```
  prob = 0.95,
  keep_raw = TRUE,
  pivot = NULL,
  quiet = FALSE,
  ...
)
```

Arguments

Y	A qsort_data object, or a J x N numeric matrix of grid positions with statements as rows and participants as columns. Every sort must obey the forced distribution exactly.
K	Integer number of factors.
iterations	First-check chain length (default 12000, the settings frozen in the accompanying paper).
burn	Burn-in iterations (default 2000), paid once; extensions continue the chain.
thin	Keep every thin-th post-burn draw (default 5).
max_iterations	Total-iteration cap for the gated extension ladder (default 48000).
seed	Optional integer seed; fits are exactly reproducible given the seed.
sigma_scale	Half-normal prior scale for the per-factor loading scales (default 1).
rhat_max, ess_min	Gate thresholds (defaults 1.01 and 400).
prob	Credible-interval probability stored on the fit (default 0.95).
keep_raw	Keep the pre-alignment draws on the fit (default TRUE); required by extend() and by realignment under a different pivot.
pivot	Optional draw index for the alignment pivot; NULL (the default) uses the median-condition-number rule.
quiet	Suppress progress messages (default FALSE).
...	Unused; supplying a removed 0.1.0 argument gives a migration error.

Value

A bayesqm_fit carrying aligned draws, the gate report, the alignment record, and (when keep_raw = TRUE) the raw draws and sampler state.

Examples

```
fit <- demo_fit()
fit
```

`fit_ladder` *Fit the model over a ladder of K*

Description

Fits `fit_bayesian()` at each K in the ladder and stores, per rung, the fit together with the adequacy and support evidence that `select_k()` consumes. Expect roughly one full fit's runtime per rung; a message up front says how many rungs are coming.

Usage

```
fit_ladder(
  Y,
  K_max = NULL,
  K_min = 2,
  q = 0.05,
  screen_draws = 30,
  screen_mixes = 60,
  quiet = FALSE,
  ...
)
```

Arguments

`Y`	As in `fit_bayesian()`.
`K_max`	Largest K to fit. `NULL` (default) uses `min(6, floor(N / 3))`, capped at 3 when `N <= 12` — with a dozen sorts or fewer, the data cannot formally discriminate adjacent K.
`K_min`	Smallest K to fit (default 2).
`q`	False-discovery level for the support evidence stored on each rung (default 0.05); `select_k()` can re-select at another q.
`screen_draws, screen_mixes`	Budget for the per-rung person check (defaults 30 and 60; the cluster signal needs no more).
`quiet`	Suppress per-rung messages (default `FALSE`).
`...`	Passed to `fit_bayesian()` (iterations, seed, ...).

Value

A `bayesqm_ladder`: the per-rung fits and evidence.

flip_factor *Flip the pole of one factor*

Description

Reverses the sign of one factor's scores and loadings in every draw. The polarity canon orients factors so defining sorts load positively; this is the judgmental override.

Usage

```
flip_factor(fit, factor)
```

Arguments

fit	A bayesqm_fit.
factor	Factor index or "f2"-style name.

Value

The fit with the factor's pole reversed.

generate_data *Simulate Q-sort data*

Description

generate_data() is the top-level data-generating function used by the package's simulation studies and tests. It builds a loading matrix (generate_loadings()), factor scores, noise of the chosen type (generate_noise()), and discretises the continuous signal onto a forced Q-sort grid (discretize_to_grid()). See also get_distribution() for the standard forced-distribution lookup.

Usage

```
generate_data(
  N,
  J,
  K,
  noise_sd = 1,
  error_type = "normal",
  nu = 5,
  contam_prop = 0.1,
  contam_scale = 4,
  loading_type = "simple",
  primary_range = c(0.55, 0.85),
```

```
  cross_range = c(-0.15, 0.15),
  seed = NULL
)

generate_loadings(
  N,
  K,
  primary_range = c(0.55, 0.85),
  cross_range = c(-0.15, 0.15),
  type = "simple"
)

generate_noise(
  J,
  N,
  type = "normal",
  sd = 1,
  nu = 5,
  contam_prop = 0.1,
  contam_scale = 4
)

discretize_to_grid(Y_cont, distr)

get_distribution(J)
```

Arguments

`N`, `J`, `K`	Numbers of participants, statements, and factors.
`noise_sd`	Residual SD.
`error_type`	One of `"normal"`, `"t"`, `"contaminated"`.
`nu`	Degrees of freedom for `error_type = "t"`.
`contam_prop`, `contam_scale`	Contamination rate and scale.
`loading_type`	`"simple"` or `"complex"`.
`primary_range`, `cross_range`	Uniform ranges for primary and cross-loadings.
`seed`	Optional RNG seed; restored on exit.
`type`	For `generate_noise()`, one of `"normal"`, `"t"`, `"contaminated"`. For `generate_loadings()`, `"simple"` or `"complex"`.
`sd`	Residual SD for `generate_noise()`.
`Y_cont`	Continuous scores (for `discretize_to_grid()`).
`distr`	Integer forced-distribution counts.

Value

`generate_data()` returns a list with `Y`, `Lambda_true`, `F_true`, `distribution`, `N`, `J`, `K`. The component helpers return their respective raw objects.

`grizzly_sorts` *Grizzly bear reintroduction Q sorts*

Description

The grizzly bear reintroduction Q dataset from Easter et al. (2025): 67 participants each force-sorted 41 statements about reintroducing grizzly bears to the North Cascades onto an eleven-column grid (quotas 1-2-3-5-6-7-6-5-3-2-1, printed values -5 to +5). A real, published panel where the two-signal rule selects a two-factor solution, used in the package documentation.

Usage

```
grizzly_sorts
```

Format

A qsort_data object: the 41 x 67 integer matrix of printed grid values with statement and participant ids, and the forced distribution.

Source

Easter, T. S., Santo, A. R., Sage, A. H., Carter, N. H., Chan, K. M. A., & Ransom, J. I. (2025). Divergent values and perspectives drive three distinct viewpoints on grizzly bear reintroduction in Washington, the United States. *People and Nature*, 7, 127–145. doi:10.1002/pan3.10748. Data from the Dryad repository under CC0, doi:10.5061/dryad.73n5tb369.

Examples

```
grizzly_sorts
plot(grizzly_sorts, participants = 1:4)
```

`import-aliases` *qmethod-style import aliases*

Description

Thin aliases that forward to `read_pqmethod()`, `read_qsort()` (HTMLQ auto-detection), `read_kenq()`, and `read_easyhtml_firebase()`. These exist only so scripts written against the `qmethod` package continue to work; new code should call the `read_*` functions directly.

Usage

```
import.pqmethod(file, ...)

import.htmlq(file, ...)

import.kenq(file, ...)

import.easyhtmlq(file, ...)
```

Arguments

file	Path to the data file.
...	Passed to the underlying reader.

Value

A qsort_data object.

---

loglik_person *Person-level partition log-likelihoods*

Description

GHK-simulated log-likelihood of each participant's sort under each kept posterior draw (thinned to draws), the input PSIS-LOO needs. The GHK seed depends only on the draw index and the person, never on K, so log-likelihoods are common-random-number comparable across a ladder of fits.

Usage

```
loglik_person(fit, draws = 100, R = 64, seed = 1)
```

Arguments

fit	A bayesqm_fit.
draws	Posterior draws to evaluate (default 100, thinned evenly).
R	GHK replications per evaluation (default 64).
seed	Base seed for the GHK draws (default 1).

Value

A draws x N matrix of log-likelihoods.

loo_ladder *PSIS-LOO across the ladder, as directional corroboration*

Description

Person-level GHK log-likelihoods per rung, fed to PSIS-LOO. Reported as directional corroboration only: with typical Q panels (N well below 100), differences in expected log predictive density between adjacent K have unreliable standard errors. The GHK seeds are common across rungs, so comparisons are common-random-number paired.

Usage

```
loo_ladder(ladder, draws = 100, R = 64)
```

Arguments

ladder	A bayesqm_ladder.
draws, R	As in loglik_person().

Value

A data frame: K, elpd_loo, se, max_pareto_k.

matchalign *MatchAlign post-processing for partition-model draws*

Description

Resolves rotational, sign, and label-permutation ambiguity in posterior draws by the MatchAlign procedure of Poworoznek et al. (2025), adapted to the partition model: varimax rotation of the loadings per draw, a pivot draw at the median condition number (overridable), pivot polarity fixed so each factor's loadings skew positive (defining sorts load positively), greedy signed matching of loading columns to the pivot, and a Procrustes rotation. A final pass re-orients every draw to the varimax of the aligned mean loadings, re-signed to positive skew, so the delivered orientation does not inherit one draw's sampling noise. One rotation is applied to loadings and scores alike, and the loading scale sigma is carried through.

Called for its side effect on the draw arrays of an engine-level fit; users normally receive already-aligned draws from fit_bayesian() and need this only to realign under a different pivot.

Usage

```
matchalign(fit, pivot = NULL)
```

Arguments

`fit`	A fit carrying draw arrays `F` (`[T, J, K]`), `Lambda` (`[T, N, K]`), `sigma` (`[T, K]`), and `K`.
`pivot`	Optional draw index to use as the alignment pivot. `NULL` (the default) selects the draw whose loading condition number sits at the median.

Value

The fit with aligned `F`, `Lambda`, and `sigma`, and the chosen `pivot` index appended.

References

Poworoznek, E., Anceschi, N., Ferrari, F., & Dunson, D. (2025). Efficiently Resolving Rotational Ambiguity in Bayesian Matrix Sampling with Matching. *Bayesian Analysis*.

`obesity_sorts` *Childhood obesity Q sorts*

Description

The childhood obesity Q dataset from Akhtar-Danesh (2023): 33 participants each force-sorted 42 statements about childhood obesity onto a nine-column grid (quotas 2-4-5-6-8-6-5-4-2, printed values -4 to +4). A real, published panel at typical Q scale, used in the package documentation.

Usage

```
obesity_sorts
```

Format

A qsort_data object: the 42 x 33 integer matrix of printed grid values with statement and participant ids, and the forced distribution.

Source

Akhtar-Danesh, N. (2023). Impact of factor rotation on Q-methodology analysis. *PLOS ONE*, 18(9), e0290728. doi:10.1371/journal.pone.0290728

Examples

```
obesity_sorts
plot(obesity_sorts, participants = 1:4)
```

plot_choice_k *The two-signal choice-of-K display*

Description

The whole decision in one strip, one row per ladder rung. The left span shows the adequacy signal, the extra-factor percentile inside its shaded band, with a warning ring where the person check found a mutual unspanned cluster. The right span shows one chip per factor, coloured when the factor is supported and grey when not, with the reason inside: `f` means fewer than two selected flags, `d` means no selected distinguishing statement. Each row ends with its own verdict in words, the selected row is boxed, and when no K passes both checks the conclusion is written across the bottom of the plot.

Usage

```
plot_choice_k(x)
```

Arguments

`x`	A `bayesqm_selection` from `select_k()`.

Value

`x`, invisibly.

plot_dist_cons *Statement contrasts between two factors*

Description

Per statement, the posterior contrast interval for one factor pair, the posterior critical difference `delta_kl` (dashed), and the grid-width consensus region (`delta_grid()`, shaded). Verdict colours: selected distinguishing listings, consensus statements, indeterminate.

Usage

```
plot_dist_cons(...)

plot_contrasts(fit, pair = 1, q = 0.05, cons_floor = 0.95)
```

Arguments

`...`	Passed on.
`fit`	A `bayesqm_fit` with `K >= 2`.
`pair`	Which factor pair to draw (index into the pair list or `"f1-f2"`-style name; default 1).
`q, cons_floor`	As in `compute_qdc()`.

Value

`fit`, invisibly.

`plot_factor_array` *The factor array on its grid*

Description

The reported array for one factor drawn as the physical sorting grid: columns are grid positions with their quota heights, each cell carries its statement, and cell shading shows how certain the placement is (the posterior probability of that statement landing in that column).

Usage

```
plot_factor_array(fit, factor = 1, labels = NULL)
```

Arguments

`fit`	A `bayesqm_fit`.
`factor`	Which factor to draw (index or `"f2"`-style name; default 1).
`labels`	Optional statement labels (defaults to the statement ids).

Value

`fit`, invisibly.

`plot_loading_posterior`
Bounded loadings with credible intervals

Description

One row per participant, one interval per factor on the bounded loading scale, with selected flags marked. The dotted rules sit at the classical descriptive cut-off `1.96 / sqrt(J)` (reference).

Usage

```
plot_loading_posterior(fit, q = 0.05)

## S3 method for class 'bayesqm_fit'
plot(x, ...)
```

Arguments

fit	A bayesqm_fit.
q	False-discovery level for the flag marks (default 0.05).
x, ...	A bayesqm_fit and arguments passed on.

Value

fit, invisibly.

---

plot_membership *Flag probabilities with the unclassified state*

Description

One row per participant, one dot per factor at its posterior flag probability (both poles combined; a minus sign marks a dominant negative pole), and a grey square for the probability of remaining unclassified. Filled dots are selected flags, and the dotted rule is the 0.5 selection floor.

Usage

```
plot_membership(...)

plot_flags(fit, q = 0.05)
```

Arguments

...	Passed on.
fit	A bayesqm_fit.
q	False-discovery level for the selection marks (default 0.05).

Value

fit, invisibly.

plot_person_check *The person check against the mixed bands*

Description

The `m` (model agreement) by `w` (person agreement) scatter with the mixed-replication bands; verdict colours, and arrows from unspanned persons to their nearest partners.

Usage

```
plot_person_check(x)
```

Arguments

`x`	A `bayesqm_persons` from `check_persons()`.

Value

`x`, invisibly.

plot_ppc *Posterior-predictive check display*

Description

The replicated distributions behind `check_fit()`: the agreement RMSE against its double-replicate reference, and the observed extra-factor eigenvalue in its replicated distribution.

Usage

```
plot_ppc(x)
```

Arguments

`x`	A `bayesqm_checks` from `check_fit()`.

Value

`x`, invisibly.

plot_sorts *Preview every participant's sort*

Description

Draws each participant's completed sort as their own pyramid: one tile per statement, stacked in its grid column, colored along the disagree-agree axis. This is the look-at-the-decisions-first view: run it straight after import, before any analysis, to see how every participant grouped the statements. `plot(qdata)` does the same.

Usage

```
plot_sorts(x, participants = NULL, per_page = 12, labels = NULL, cex = NULL)
```

Arguments

`x`	A `qsort_data` object or a `J x N` matrix of sorts.
`participants`	Which participants to draw (names or indices); `NULL` (default) draws everyone.
`per_page`	How many pyramids per page (default up to 12, arranged automatically); further participants continue on the next page.
`labels`	Optional statement labels for the tiles (defaults to the statement ids).
`cex`	Tile label size; `NULL` (default) adapts to the tallest column.

Value

The input, invisibly.

Examples

```
distr <- c(1, 2, 3, 2, 1)
set.seed(1)
Y <- replicate(4, {
  r <- integer(9)
  r[order(rnorm(9))] <- rep(seq_along(distr), distr)
  r - 3                                  # printed labels -2..+2
})
plot_sorts(qsort_data(Y, distribution = distr, validate = FALSE))
```

plot_statement *One statement, in depth*

Description

The posterior of a single statement's score under each factor, drawn as full densities with the medians marked, plus the statement's verdict and its pairwise contrasts in the margin text. The drill-down for the statement a write-up centers on.

Usage

```
plot_statement(fit, statement, q = 0.05, cons_floor = 0.95)

plot_zscore_posterior(...)
```

Arguments

`fit`	A `bayesqm_fit`.
`statement`	Statement id or index.
`q, cons_floor`	As in `compute_qdc()`.
`...`	Passed on.

Value

`fit`, invisibly.

plot_tucker *Convergence and alignment view*

Description

Traces of the invariant person spreads `s_i` for the persons with the lowest effective sample size — the series the convergence gate reads — plus the per-draw congruence of the aligned draws to the pivot, with the gate report in the margin. A congruence histogram hugging 1 says the alignment found one common orientation; a long left tail says some draws sit far from it.

Usage

```
plot_tucker(...)

plot_convergence(fit, n_series = 3)
```

Arguments

`...`	Passed on.
`fit`	A `bayesqm_fit`.
`n_series`	How many worst-ESS persons to trace (default 3).

Value

`fit`, invisibly.

`plot_zscores` *Statement scores across factors, whole panel*

Description

Every statement's score under every factor, with nested 50% and 95% credible intervals — the classic Q z-score chart carrying its uncertainty. Statements are ordered by how strongly the factors diverge on them, so the top of the chart is where the viewpoints disagree and the bottom is where they concur; the left margin marks each statement's verdict.

Usage

```
plot_zscores(
  fit,
  order_by = c("divergence", "score"),
  q = 0.05,
  cons_floor = 0.95
)
```

Arguments

`fit`	A `bayesqm_fit`.
`order_by`	`"divergence"` (default; by the largest median pairwise contrast) or `"score"` (by the first factor's median score).
`q`, `cons_floor`	As in `compute_qdc()` (used for the verdict marks when `K >= 2`).

Value

`fit`, invisibly.

posterior_interval *Posterior interval generic*

Description

Generic for posterior credible intervals, defined here so the package needs no Stan-ecosystem dependency. Compatible with the generic of the same name in **rstantools**.

Usage

```
posterior_interval(object, ...)
```

Arguments

object	A fitted model object.
...	Passed to methods.

Value

See the method documentation.

posterior_interval.bayesqm_fit *Credible intervals for bayesqm_fit parameters*

Description

Posterior credible intervals for any subset of parameters in a `bayesqm_fit`. Method for `posterior_interval()`.

Usage

```
## S3 method for class 'bayesqm_fit'
posterior_interval(object, prob = 0.95, pars = NULL, regex_pars = NULL, ...)
```

Arguments

object	A `bayesqm_fit`.
prob	Coverage probability (default 0.95).
pars	Optional character vector of exact parameter names.
regex_pars	Optional regex; matching parameter names are included in addition to those named in `pars`.
...	Unused.

Value

A matrix with one row per parameter and two columns for the lower and upper interval bounds.

prior_summary *Prior summary generic*

Description

Generic for summarizing the priors a model was fit with, defined here so the package needs no Stan-ecosystem dependency. Compatible with the generic of the same name in **rstantools**.

Usage

```
prior_summary(object, ...)
```

Arguments

object	A fitted model object.
...	Passed to methods.

Value

See the method documentation.

prior_summary.bayesqm_fit *Prior summary for a bayesqm_fit*

Description

The partition model's priors, as a printable `bayesqm_prior` object. Method for `prior_summary()`.

Usage

```
## S3 method for class 'bayesqm_fit'
prior_summary(object, ...)
```

Arguments

object	A `bayesqm_fit`.
...	Unused.

Value

A `bayesqm_prior` data frame with columns `parameter` and `prior`.

qsort_data *Construct a validated qsort_data object*

Description

`qsort_data()` is the canonical constructor for a Q-sort dataset. `validate_qsort()`, `check_distribution()`, and `infer_distribution()` are the validation helpers used internally by the constructor and by the file readers. `parse_distribution()` accepts a numeric vector, a comma-/semicolon-/space-separated string, or a text file containing one of those.

Usage

```
qsort_data(
  Y,
  statements = NULL,
  participants = NULL,
  distribution = NULL,
  metadata = list(),
  source = "manual",
  validate = TRUE
)

validate_qsort(qdata, distribution = NULL)

check_distribution(Y, distribution)

infer_distribution(Y)

parse_distribution(x)
```

Arguments

`Y`	A `J x N` numeric matrix (statements as rows, participants as columns) or a data frame.
`statements, participants`	Optional character vectors of IDs; default to `S1..SJ` and `P1..PN`.
`distribution`	Optional integer vector of forced-distribution counts. Inferred from `Y[, 1]` when `NULL`.
`metadata`	Optional named list of study-level info.
`source`	Provenance string stored on the object.
`validate`	If `TRUE` (default), run `validate_qsort()` and emit warnings / messages for any issues found.
`qdata`	A `qsort_data` object or bare matrix, passed to `validate_qsort()`.
`x`	Numeric vector, character string, or path to a file containing the forced distribution, passed to `parse_distribution()`.

Value

qsort_data() returns a qsort_data S3 list with fields Y, statements, participants, distribution, metadata, and source. validate_qsort() returns a list with valid, issues, warnings, and summary. check_distribution() returns a list with ok, non_conforming, and grid_values. infer_distribution() and parse_distribution() return integer vectors.

---

qsort_data-methods *Print, summary, and matrix conversion for qsort_data*

Description

Print, summary, and matrix conversion for qsort_data

Usage

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

## S3 method for class 'qsort_data'
summary(object, ...)

## S3 method for class 'qsort_data'
as.matrix(x, ...)
```

Arguments

x, object	A qsort_data object.
...	Unused.

Value

print() and summary() return the input invisibly; as.matrix() returns the J x N Q-sort matrix.

---

read_qsort *Read Q-sort data from file*

Description

read_qsort() auto-detects the file format from extension and content and dispatches to a specialised reader. The specialised readers are also exported for explicit use:

- read_qsort_csv(), read_qsort_excel() for generic CSV / Excel (with HTMLQ / FlashQ / Ken-Q auto-detection baked in)
- read_pqmethod() for PQMethod .DAT files

- `read_kenq()` for Ken-Q JSON or CSV
- `read_kenq_excel()` for multi-sheet Ken-Q Excel (Type 1 and Type 2, both old and Ver2 sub-formats)
- `read_kade_zip()` for KADE ZIP archives
- `read_easyhtml_firebase()` for Easy-HTMLQ Firebase JSON
- `read_statements()` for a standalone statement-text file

All readers return a `qsort_data` object in `J x N` orientation (statements as rows, participants as columns).

Usage

```
read_qsort(file, format = "auto", ...)

read_qsort_csv(
  file,
  orientation = c("auto", "statements_rows", "participants_rows"),
  id_col = c("auto", "first", "none"),
  statements = NULL,
  distribution = NULL,
  ...
)

read_qsort_excel(
  file,
  sheet = 1,
  orientation = c("auto", "statements_rows", "participants_rows"),
  id_col = c("auto", "first", "none"),
  statements = NULL,
  distribution = NULL,
  ...
)

read_pqmethod(file, statements_file = NULL)

read_kenq(file, format = c("auto", "json", "csv"))

read_kenq_excel(file)

read_kade_zip(file)

read_easyhtml_firebase(file)

read_statements(file, column = 1, id_column = NULL)
```

Arguments

`file`	Path to the data file.

format	For read_qsort(), "auto" (default) or one of "csv", "excel", "pqmethod", "kenq", "kenq_excel", "kade", "easyhtml_firebase". For read_kenq(), one of "auto", "json", "csv".
...	Passed to the underlying reader.
orientation	For generic CSV/Excel: "auto", "statements_rows", or "participants_rows".
id_col	For generic CSV/Excel: "auto", "first", or "none".
statements, distribution	Optional overrides passed to qsort_data().
sheet	Excel sheet name or index (default 1).
statements_file	For PQMethod, optional separate statements file.
column, id_column	For read_statements(): column index or name.

Value

A qsort_data object, except read_statements() which returns a named character vector.

---

rename_factors *Rename the factors*

Description

Replaces the f1 ... fK labels with substantive names everywhere the fit carries them, so every downstream table and plot uses the new names.

Usage

```
rename_factors(fit, new_names)
```

Arguments

fit	A bayesqm_fit.
new_names	Character vector of length K.

Value

The renamed fit.

rotate_factors *Rotate every aligned draw toward a target*

Description

Judgmental rotation: each aligned draw's statement scores are rotated toward an analyst-specified target by Procrustes, and the loadings receive the same rotation (its inverse transpose when `oblique = TRUE`), so the latent utilities every draw implies are unchanged. Summaries computed from the returned fit are summaries of the rotated solution, with full posterior uncertainty.

Usage

```
rotate_factors(fit, target, oblique = FALSE)
```

Arguments

`fit`	A `bayesqm_fit`.
`target`	A `J x K` numeric matrix of target scores (for example, a hand-edited copy of the posterior-mean scores).
`oblique`	Allow an oblique (non-orthogonal) rotation (default `FALSE`).

Value

The fit with rotated draws; `fit$align$rotated` records the call.

save_bayesqm_plot *Save a bayesqm plot to file*

Description

Opens a graphics device chosen from the file extension (`.pdf`, `.svg`, `.png`, `.tiff`, `.jpeg`), evaluates `expr` so whatever it draws lands on that device, and closes the device. `expr` is lazily evaluated, so a call like `save_bayesqm_plot("fig.pdf", plot(fit))` does not draw to the current screen device first. If `expr` returns a `ggplot` object (for example, from `ggplot2::autoplot()`), it is `print()`ed onto the device.

Usage

```
save_bayesqm_plot(file, expr, width = 7.2, height = 5, dpi = 300)
```

Arguments

`file`	Output path. Extension determines the device.
`expr`	Plotting expression (lazily evaluated).
`width, height`	Dimensions in inches. Defaults to a 7.2 x 5 inch two-column journal figure.
`dpi`	Resolution for raster formats (`png`, `tiff`, `jpeg`).

Value

The `file` path, invisibly.

Examples

```
f <- file.path(tempdir(), "fig.pdf")
save_bayesqm_plot(f, plot(1:10))
unlink(f)
```

`select_k` *Choose K by the two-signal rule*

Description

Signal one is posterior-predictive adequacy: the extra-factor check sits inside the central band and the person check shows no mutual unspanned cluster. Signal two is parsimony: every factor is supported, meaning at least two selected flags and at least one selected distinguishing listing. The selection is the smallest adequate K with all factors supported. When no factor is supported at any rung the verdict separates two opposite situations: a panel sharing nothing (no factor ever attracts two flags) and a panel so unanimous that flags abound but no statement distinguishes any pair — a single shared viewpoint, reported as such. When adequacy and support never coincide, both candidate solutions are reported rather than forcing a winner.

Usage

```
select_k(ladder, q = NULL, band = c(0.05, 0.95))
```

Arguments

`ladder`	A `bayesqm_ladder`.
`q`	Re-select the support evidence at this level; `NULL` (the default) keeps the level the ladder stored.
`band`	Central adequacy band for the extra-factor percentile (default `c(0.05, 0.95)`).

Value

A `bayesqm_selection` with the verdict, the per-rung evidence table, and the selected `K` (or `NA`).

sigma.bayesqm_fit *Posterior-mean loading scales*

Description

Posterior-mean loading scales

Usage

```
## S3 method for class 'bayesqm_fit'
sigma(object, ...)
```

Arguments

object	A bayesqm_fit.
...	Unused.

Value

Named numeric vector of posterior-mean sigma_k.

tucker_congruence *Tucker's congruence and orthogonal Procrustes rotation*

Description

Linear-algebra utilities shared by MatchAlign and the recovery assessments. tucker_congruence(x, y) computes sum(x*y) / sqrt(sum(x^2) * sum(y^2)). procrustes_rotation(X, Target) finds the orthogonal R minimising ||X R - Target||_F and returns X %*% R with R attached as attribute "rotation".

Usage

```
tucker_congruence(x, y)

procrustes_rotation(X, Target)
```

Arguments

x, y	Numeric vectors (for Tucker).
X, Target	Matrices (for Procrustes).

Value

Tucker returns a scalar; Procrustes returns the rotated matrix with a "rotation" attribute.

Index