Package: bayesqm (via r-universe)

June 17, 2026

Type Package

Title Bayesian Q Methodology: Probabilistic Factor Analysis

Version 0.1.0

Date 2026-05-18

Description A Bayesian factor-analytic framework for Q methodology. Fits a low-rank factor model to Q-sort data with a Student-t likelihood and a hierarchical normal prior on loadings, samples the posterior with Stan, resolves rotational ambiguity via the MatchAlign post-processing of Poworoznek et al. (2025) <doi:10.1214/25-BA1544>, and returns posterior summaries including credible intervals for loadings and factor scores, probabilistic dominant-factor membership, distinguishing and consensus statements, and PSIS-LOO-based factor enumeration following Vehtari et al. (2017) <doi:10.1007/s11222-016-9696-4> with the Sivula et al. (2025) <doi:10.1214/25-BA1569> parsimony rule.

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

Config/testthat/edition 3

Depends R (>= 4.1.0)

Imports stats, utils, tools, parallel, rstantools (>= 2.3.0)

Suggests cmdstanr (>= 0.8.0), rstan (>= 2.32.0), loo (>= 2.7.0), GPArotation (>= 2024.3-1), lpSolve, readxl (>= 1.4.0), jsonlite (>= 1.8.0), posterior (>= 1.5.0), ggplot2 (>= 3.4.0), ggdist (>= 3.3.0), ggridges (>= 0.5.0), scales (>= 1.2.0), knitr (>= 1.40), rmarkdown (>= 2.20), testthat (>= 3.0.0), vdiffr (>= 1.0.0)

VignetteBuilder knitr

Additional_repositories https://stan-dev.r-universe.dev

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Raymond Dacosta Azadda [aut, cre], AK-ACE Team [aut], Karsten Hueffer [aut], Taa'aii Peter [aut], Stacy Rasmus [aut]

Maintainer Raymond Dacosta Azadda <rdazadda@alaska.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-17 19:33:24 UTC

RemoteUrl https://github.com/cran/bayesqm

RemoteRef HEAD

RemoteSha 9ea57188d3fe62f5e2957cd2bfd379c73bc0a5f0

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 `"blue"` (default), `"teal"`, `"red"`, `"purple"`, and `"grey"`. For full control, pass a named list with slots `light`, `mid`, `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()
fit <- demo_fit(N = 6, J = 10, K = 2, Td = 50, seed = 1)
bayesqm_set_colors("teal")
plot(fit)
bayesqm_set_colors("blue")  # restore default
```

bayesqm-fit-accessors *Standard R accessors for bayesqm_fit*

Description

S3 methods that make a `bayesqm_fit` behave like a standard R modelling object: `coef()` returns the posterior-mean loadings, `fitted()` the posterior-mean fitted Y on the original Q-sort scale, `residuals()` is `Y - fitted(fit)`, `sigma()` is the posterior-mean residual scale, `nobs()` is the number of participants, and `family()` returns a small `bayesqm_family` list with `$family`, `$link`, and `$nu`. `as.matrix()`, `as.array()`, and `as.data.frame()` return the posterior draws in Stan-style parameter naming (`Lambda[i,k]`, `F[j,k]`, `nu`, `sigma`, `tau`), which the **posterior**, **bayesplot**, and **tidybayes** packages consume natively.

`update()` re-fits the model with modified arguments; the original call and stored data are reused.

Usage

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

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

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

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

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

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

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

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

## S3 method for class 'bayesqm_fit'
as.array(x, ...)

## S3 method for class 'bayesqm_fit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'bayesqm_fit'
update(object, ..., evaluate = TRUE)
```

Arguments

object, x	A bayesqm_fit object.
...	Further arguments (e.g. arguments for update()).
row.names, optional	Passed to as.data.frame().
evaluate	If FALSE, update() returns the modified call without evaluating it.

Value

Depends on the method; see Description.

bayesqm-fit-methods *Print and summary methods for bayesqm_fit and bayesqm_run*

Description

`print()` shows a compact, brms-style header with convergence and the first few loadings. `summary()` expands with factor characteristics, the PSIS-LOO estimate, the divergence summary, and the MatchAlign Tucker-phi diagnostic. Both methods exist for `bayesqm_fit` (returned by `fit_bayesian()`) and `bayesqm_run` (returned by `run_bayes()`).

Usage

```
## S3 method for class 'bayesqm_fit'
print(x, digits = 2, length = 10, ...)

## S3 method for class 'bayesqm_fit'
summary(object, digits = 3, ...)

## S3 method for class 'bayesqm_run'
print(x, digits = 2, ...)

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

Arguments

`x, object`	A `bayesqm_fit` or `bayesqm_run` object.
`digits`	Number of digits to print.
`length`	Maximum number of participant rows to show in the compact loading table.
`...`	Unused.

Value

The input, invisibly.

bayesqm-membership *Probabilistic factor-membership and divergence summaries*

Description

Posterior summaries of factor membership and statement interpretation, each computed entirely from posterior draws:

- `compute_threshold_prob()` returns the `N x K` posterior probability that `|lambda_ik| > threshold`, i.e. the Bayesian version of the Brown (1980) flagging rule.
- `compute_dominant_prob()` returns the `N x K` posterior probability that factor `k` is the dominant factor for participant `i`.
- `compute_dominant_sign()` returns the length-`N` posterior probability that the dominant loading is positive; participants with probability below 0.5 are negative exemplars.
- `compute_divergence()` returns the posterior of the viewpoint divergence `D_j` for every statement, together with the distinguishing and consensus probabilities it implies.
- `classify_membership()` turns dominant probabilities into a per-participant descriptive tier (Strong / Moderate / Weak).

Usage

```
compute_threshold_prob(Lambda_draws, threshold)

compute_dominant_prob(Lambda_draws)

compute_dominant_sign(Lambda_draws)

compute_divergence(F_draws, delta = NULL, delta_grid = NULL)

classify_membership(Lambda_draws, strong = 0.8, moderate = 0.6)
```

Arguments

`Lambda_draws`	Array of shape `[T, N, K]` of aligned loading draws.
`threshold`	Numeric threshold; a natural default is `1.96 / sqrt(J)` for the Brown rule.
`F_draws`	Array of shape `[T, J, K]` of aligned factor-score draws.
`delta`	Substantive separation for the distinguishing and consensus probabilities. The fit pipeline supplies the default, the reliability-adjusted critical difference of `critical_delta()`; `suggest_delta()` is an alternative. If `NULL` the `D_j` posterior (median, 95% CrI, `g_jk`) is still returned but the distinguishing/consensus probabilities are `NA`.
`delta_grid`	Optional numeric vector of `delta` values for a sensitivity sweep.
`strong, moderate`	Tier cutoffs on `max P(dominant)` (defaults 0.80 and 0.60).

Details

For statement `j` with standardized viewpoint scores `f_{j1}, ..., f_{jK}`, the divergence estimand is the mean absolute pairwise difference `D_j = 2 / (K (K - 1)) * sum_{k < l} |f_jk - f_jl|`.

The mean is used rather than the maximum, which is an order statistic over the K (K - 1) / 2 contrasts and is inflated when the posterior is diffuse. compute_divergence() reports the posterior median and central 95% credible interval of D_j, pi_D = P(D_j > delta | Y) (distinguishing) and pi_C = P(D_j < delta | Y) = 1 - pi_D (consensus), and the per-viewpoint departure g_jk = f_jk - mean_{l != k} f_jl with its dominant viewpoint, sign, and P(|g_jk| > delta | Y). The probabilities are the reported quantities; no fixed probability cutoff defines a distinguishing or consensus statement.

Value

compute_threshold_prob() and compute_dominant_prob() return N x K matrices. compute_dominant_sign() returns a length-N named vector. compute_divergence() returns a list (see Details). classify_membership() returns a data frame.

---

caption_bayesqm *Dynamic figure caption for a bayesqm_fit*

Description

Returns a human-readable caption string summarising the model configuration (K, N, J, family), the sampler (backend, chains, post-warmup draws), the interval probability, and convergence diagnostics (max Rhat, divergent transitions).

Usage

```
caption_bayesqm(fit, include_ref = TRUE, include_diag = TRUE)
```

Arguments

fit	A bayesqm_fit.
include_ref	Logical; append a brief package-attribution line.
include_diag	Logical; append the convergence-diagnostic line.

Value

A length-1 character string.

Examples

```
fit <- demo_fit(N = 6, J = 10, K = 2, Td = 50, seed = 1)
cat(caption_bayesqm(fit))
```

compute_factor_array *Factor arrays on the forced Q-sort distribution*

Description

Posterior-mean factor z-scores ranked onto the study's forced distribution. The result is a tidy data frame with one row per statement and per-factor integer grid scores. For the continuous z-scores with credible intervals, use `compute_zscores()`.

Usage

```
compute_factor_array(F_draws, Y)
```

Arguments

`F_draws`	Array of shape `[T, J, K]` of factor-score draws.
`Y`	The Q-sort matrix whose first column supplies the forced distribution (as in the original study's grid).

Value

A data frame with columns `statement` and `f1_grid, f2_grid, ..., fK_grid`.

compute_loadings *Posterior summary of participant factor loadings*

Description

Posterior mean and central credible-interval bounds for each participant-factor loading, returned as a tidy data frame with one row per participant and three columns per factor.

Usage

```
compute_loadings(Lambda_draws, prob = 0.95)
```

Arguments

`Lambda_draws`	Array of shape `[T, N, K]` of MatchAlign-aligned loading draws (e.g. `fit$Lambda_draws`).
`prob`	Coverage probability for the credible interval (default 0.95).

Value

A data frame with columns `participant`, and three numeric columns per factor: `fk_loa` (posterior mean), `fk_lower`, and `fk_upper`, for `k = 1..K`.

compute_posterior_scalars

Posterior summary of scalar hyperparameters

Description

Returns a tidy data frame with one row per scalar parameter and columns `mean`, `median`, `sd`, `lower`, `upper`.

Usage

```
compute_posterior_scalars(scalar_draws, prob = 0.95)
```

Arguments

`scalar_draws`	Named list of draw vectors. For a `bayesqm_fit`, pass `fit$hyperparams` directly.
`prob`	Coverage probability for the credible interval.

Value

A data frame.

compute_zscores

Posterior summary of statement factor z-scores

Description

Posterior mean and central credible-interval bounds for each statement-factor z-score, returned as a tidy data frame with one row per statement and three columns per factor.

Usage

```
compute_zscores(F_draws, prob = 0.95)
```

Arguments

`F_draws`	Array of shape `[T, J, K]` of MatchAlign-aligned factor-score draws (e.g. `fit$F_draws`).
`prob`	Coverage probability for the credible interval.

Value

A data frame with columns `statement`, and three numeric columns per factor: `fk_zsc` (posterior mean), `fk_lower`, and `fk_upper`, for `k = 1..K`.

critical_delta *Reliability-adjusted critical difference (default* delta*)*

Description

The default separation for the distinguishing and consensus probabilities: the reliability-adjusted critical difference used to flag distinguishing statements in classical Q analysis (Brown 1980; Zabala & Pascual 2016), here generalized by computing it from the posterior dominant-factor counts. With `p_f` the number of participants whose posterior dominant factor is `f`, `r_f = p_f r0 / (1 + (p_f - 1) r0)` and `SE_f = sqrt(1 - r_f)`, `delta = z * mean_{k < l} sqrt(SE_k^2 + SE_l^2)`. The reliability is the stable population reliability of the design, not the posterior estimation spread.

Usage

```
critical_delta(Lambda_draws, level = 0.05, r0 = 0.8)
```

Arguments

`Lambda_draws`	Array of shape `[T, N, K]` of aligned loading draws.
`level`	Two-sided level for the critical value, `z = qnorm(1 - level/2)`. `0.05` (default) gives `z = 1.96`; `0.01` gives `z = 2.58`. Report sensitivity over `level`.
`r0`	Conventional single-sort reliability (default `0.80`; Brown 1980).

Value

A single numeric value, the critical-difference `delta`.

Examples

```
critical_delta(array(rnorm(200 * 8 * 3), c(200, 8, 3)))
```

demo_fit *A synthetic bayesqm_fit for examples and tutorials*

Description

Returns a `bayesqm_fit` with realistic Q-methodology structure: every participant has a dominant factor, roughly 40 percent of the statements polarise the factor pair, 10 percent are consensus, and the remainder are weakly partial. Use it for documentation, teaching materials, and the package vignette; it is not a substitute for `fit_bayesian()` on real data.

Usage

```
demo_fit(N = 20, J = 22, K = 2, Td = 400, seed = 1L)
```

Arguments

N	Number of participants.
J	Number of statements.
K	Number of factors.
Td	Number of posterior draws.
seed	Integer seed for reproducibility; NULL leaves the random number generator untouched.

Value

A bayesqm_fit.

Examples

```
fit <- demo_fit(N = 12, J = 15, K = 2)
plot(fit)
summary(fit)
```

---

demo_run *A synthetic bayesqm_run for examples and tutorials*

Description

Returns a bayesqm_run object carrying a plausible ELPD trajectory across K = 1..K_max, with user-chosen peak K, Sivula K, and case label. Use it to demonstrate run_bayes() output and plot_elpd() without a Stan backend; it is not a substitute for run_bayes() on real data.

Usage

```
demo_run(
  K_max = 4L,
  k_peak = 3L,
  k_sivula = 2L,
  case = c("gap", "agree", "reversed"),
  seed = 1L
)
```

Arguments

K_max	Largest K in the comparison (default 4).
k_peak	K value where ELPD peaks (default 3).
k_sivula	K chosen by the Sivula parsimony rule (default 2).
case	Case label: "agree", "gap", or "reversed".
seed	Integer seed for reproducibility; NULL leaves the random number generator untouched.

Value

A bayesqm_run.

Examples

```
run <- demo_run()
run
plot_elpd(run)
```

---

fit_bayesian *Fit a Bayesian Q-methodology factor model*

Description

Fits the low-rank Bayesian factor model to Q-sort data. Samples the posterior with Stan (via **cmdstanr** or **rstan**), resolves rotational ambiguity with MatchAlign, and returns a classed bayesqm_fit object carrying posterior-mean loadings and factor scores, credible intervals, raw draws, LOO, PPC, and diagnostics.

Usage

```
fit_bayesian(
  Y,
  K,
  stan_dir = NULL,
  robust = TRUE,
  nu = "estimate",
  chains = 4,
  iter = 2000,
  warmup = 1000,
  seed = NULL,
  adapt_delta = 0.9,
  max_draws = 2000,
  prior_loading_scale = 1,
  prior_sigma_scale = 1,
  prior_nu_alpha = 2,
  prior_nu_beta = 0.1,
  use_half_cauchy = FALSE,
  prob = 0.95,
  delta = NULL
)
```

Arguments

Y	Either a qsort_data object or a J x N numeric matrix with statements as rows and participants as columns.

K	Integer number of factors to extract.
stan_dir	Directory containing stan/factor_model.stan. NULL (the default) uses the copy shipped in inst/stan/.
robust	Logical; TRUE uses a Student-t likelihood, FALSE uses Normal.
nu	Either "estimate" (default) to sample the Student-t degrees of freedom, or a numeric value (e.g. 5, Inf) to fix it.
chains, iter, warmup	NUTS sampler settings.
seed	Optional integer seed for reproducibility.
adapt_delta	NUTS adapt_delta target (default 0.90).
max_draws	Thin post-warmup draws to at most this many before MatchAlign (default 2000).
prior_loading_scale, prior_sigma_scale, prior_nu_alpha, prior_nu_beta, use_half_cauchy	Prior hyperparameters (see the Stan model for parameterization).
prob	Credible-interval probability stored on the fit (default 0.95).
delta	Substantive viewpoint separation for the distinguishing/consensus probabilities. If NULL (default) it is computed as the reliability-adjusted critical difference (critical_delta()); pass a numeric value to override, or use suggest_delta() as an alternative.

Value

A bayesqm_fit object. See bayesqm-fit-methods for print() and summary(), and coef.bayesqm_fit() for the standard R accessors.

References

Poworoznek et al. (2025). Efficiently Resolving Rotational Ambiguity in Bayesian Matrix Sampling with Matching. *Bayesian Analysis*.

Examples

```
# Needs a working Stan backend; skipped when cmdstanr/CmdStan is absent.
has_stan <- requireNamespace("cmdstanr", quietly = TRUE) &&
  !inherits(try(cmdstanr::cmdstan_path(), silent = TRUE), "try-error")
if (has_stan) {
  sim <- generate_data(N = 8, J = 12, K = 2, seed = 1)
  fit <- fit_bayesian(sim$Y, K = 2, chains = 1, iter = 600, warmup = 300)
  summary(fit)
}
```

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.

---

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.

---

make_dominant_panel *Probabilistic dominant-factor panel*

Description

Draws a blue-gradient heatmap of P(dominant factor = k) per participant, with a right-hand "Assignment" strip (orange-red gradient) showing the verdict "Strong / Mod. / Weak F-k" for each participant.

Usage

```
make_dominant_panel(fit, title = NULL, anonymize = TRUE)
```

Arguments

fit	A bayesqm_fit.
title	Optional panel title.
anonymize	Logical; when TRUE, participants are relabelled P01..PNN.

Value

A ggplot object.

---

make_elpd_diff *Delta-ELPD plot with Sivula band, peak, and adopted-K annotations*

Description

Draws ΔELPD against K with +/- 1.96 SE whiskers, shades the Sivula rejection band ($|\Delta\text{ELPD}| < 4$), and marks the Sivula-selected K (red triangle), the ELPD peak (blue square), and an optional adopted K (orange diamond).

Usage

```
make_elpd_diff(run, title = NULL, adopted = NULL)
```

Arguments

run	A bayesqm_run object.
title	Optional panel title.
adopted	Integer K adopted by the analyst. Marked with the orange diamond. NULL suppresses this annotation.

Value

A ggplot object.

---

make_ppc_ridge *Posterior predictive RMSE ridgeline across K*

Description

Draws a ridgeline density of the posterior predictive correlation-matrix RMSE ($RMSE_R$) at every K in run, with a median tick per ridge.

Usage

```
make_ppc_ridge(run, title = NULL)
```

Arguments

run	A bayesqm_run.
title	Optional panel title.

Value

A ggplot object.

---

matchalign *MatchAlign post-processing for Bayesian factor draws*

Description

Resolves rotational, sign, and label-permutation ambiguity in posterior draws of a factor model by the three-step MatchAlign procedure of Poworoznek et al. (2025): varimax rotation per draw, median-condition pivot selection, greedy L2 signed-permutation matching, and Procrustes rotation.

Usage

```
matchalign(Lambda_draws, F_draws)
```

Arguments

`Lambda_draws`	Array of shape `[T, N, K]` of loading draws.
`F_draws`	Array of shape `[T, J, K]` of factor-score draws.

Value

A list with aligned `Lambda` and `Fmat` arrays, a `congruence` matrix of per-draw Tucker-phi per factor, and the `pivot` index used.

References

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

`plot.bayesqm_fit` *Factor-score dotchart for a bayesqm_fit*

Description

One panel per factor: horizontal dotchart of every statement's posterior median z-score with nested 50 percent (thick) and `prob`-coverage (thin) credible-interval whiskers. Statements are sorted once – by default, by the first factor's posterior mean – and that ordering is reused across panels so the reader can scan horizontally to compare factors. For a loadings-only view, see `plot_loading_posterior()`; for a single statement across factors, see `plot_zscore_posterior()`.

Usage

```
## S3 method for class 'bayesqm_fit'
plot(x, sort_by = 1L, prob = NULL, cex.lab = 0.7, ...)
```

Arguments

`x`	A `bayesqm_fit`.
`sort_by`	Integer factor index whose posterior mean is used to sort rows (default 1).
`prob`	Outer-interval coverage probability; defaults to `fit$brief$prob`.
`cex.lab`	Axis-label text size.
`...`	Additional arguments forwarded to `plot()`.

Value

The input, invisibly.

plot_dist_cons *Distinguishing/consensus divergence forest*

Description

Horizontal dot-and-whisker plot of the posterior viewpoint divergence `D_j` for every statement: posterior median with a 95% credible-interval whisker, statements ordered by the distinguishing probability `P(D_j > delta | Y)`. A dashed rule marks the substantive separation `delta` and each point is shaded by `P(D_j > delta | Y)`. By default the divergence summary stored on the fit is used (computed at the fit's `delta`); pass `delta` to recompute at a different separation without refitting.

Usage

```
plot_dist_cons(fit, delta = NULL, ...)
```

Arguments

`fit`	A `bayesqm_fit`.
`delta`	Optional separation override. If `NULL` (default) the fit's stored summary (`fit$qdc`, at `fit$brief$delta`) is used; that default is the Bayesian reliability-adjusted critical difference (`critical_delta()`). Pass a numeric value to recompute the divergence summary at a different separation.
`...`	Additional arguments forwarded to `plot()`.

Value

The input, invisibly.

plot_elpd *ELPD across K with peak and Sivula annotations*

Description

ELPD against K with +/- 1.96 SE whiskers, a solid vertical rule at the ELPD peak, and a dashed rule at the Sivula (parsimony) K. Both rules are drawn in the primary blue, distinguished by line type and by text annotations at the top of the axis. The title reports the peak-plus-Sivula case (`agree`, `gap`, `reversed`).

Usage

```
plot_elpd(run, ...)
```

Arguments

`run`	A `bayesqm_run`.
`...`	Additional arguments forwarded to `plot()`.

Value

The input, invisibly.

`plot_hyper` *Hyperparameter posterior densities*

Description

One panel per scalar hyperparameter (default `nu`, `sigma`, `tau`) showing a two-tone kernel density: the full posterior in the light shade, the central credible interval re-shaded in the mid shade, and the posterior median marked by a short tick at the baseline. Coverage defaults to `fit$brief$prob`. When a parameter's support spans more than one order of magnitude (`max / min > 20`, all positive), the x-axis is automatically rendered on a log scale so the density shape is not crushed against the lower bound.

Usage

```
plot_hyper(fit, pars = c("nu", "sigma", "tau"), log = NULL, ...)
```

Arguments

`fit`	A `bayesqm_fit`.
`pars`	Character vector of hyperparameter names to show.
`log`	`NULL` (default) auto-detects per parameter; set to `"x"` to force log, or `""` to force linear.
`...`	Additional arguments forwarded to `plot()`.

Value

The input, invisibly.

`plot_loading_posterior`
Loading forest with 50 and 95 percent credible intervals

Description

Horizontal dotchart of every participant's loading, one panel per factor, ranked by posterior mean. Each loading is drawn as a median point with nested 50 percent (thick) and 95 percent (thin) credible-interval whiskers. A faint grey vertical rule marks Brown's descriptive cut-off `+/- 1.96 / sqrt(J)`. When `highlight_flagged = TRUE`, participants in `fit$flagged[, k]` are drawn as filled points in the accent colour.

Usage

```
plot_loading_posterior(fit, factors = NULL, highlight_flagged = TRUE, ...)
```

Arguments

fit	A bayesqm_fit.
factors	Optional subset of factors to show (integer or name).
highlight_flagged	Logical; fill flagged participants.
...	Additional arguments forwarded to plot().

Value

The input, invisibly.

---

plot_membership *Dominant-factor posterior-probability heatmap*

Description

Tiled heatmap of P(argmax_k |Lambda[i, k]| = k) with one row per participant, one column per factor, rendered on a sequential blue ramp. A right-side tier strip encodes Strong / Moderate / Weak membership (per classify_membership()). A horizontal colourbar under the plot gives the probability scale. Rows are sorted by dominant factor first, then tier, then probability – so the block structure a reader wants to see is preserved.

Usage

```
plot_membership(fit, sort = TRUE, ...)
```

Arguments

fit	A bayesqm_fit.
sort	Logical; apply the default ordering.
...	Additional arguments forwarded to image().

Value

The input, invisibly.

plot_ppc *Posterior predictive check on the correlation-matrix RMSE*

Description

Histogram of the replicated correlation-matrix RMSE stored on `fit$ppc$rmse.r`: per draw, the RMSE between `cor(Y_rep)` and `cor(Y_obs)`. Rendered in the bayesplot-idiom (filled bars, no border, suppressed y-axis) with the median and central credible interval marked.

Usage

```
plot_ppc(fit, breaks = 30, ...)
```

Arguments

`fit`	A `bayesqm_fit`.
`breaks`	Passed to `hist()`.
`...`	Additional arguments forwarded to `hist()`.

Value

The input, invisibly.

plot_tucker *MatchAlign Tucker's phi distribution by factor*

Description

Boxplot of the per-draw Tucker's phi between each aligned loading column and the MatchAlign pivot, stored on `fit$align_info$congruence`. A semi-transparent strip of the individual draws is overlaid so bimodality – the visible signature of residual label-switching – is not hidden by the box. A dashed rule at 0.95 marks the conventional near-identity threshold.

Usage

```
plot_tucker(fit, ...)
```

Arguments

`fit`	A `bayesqm_fit`.
`...`	Additional arguments forwarded to `boxplot()`.

Value

The input, invisibly.

plot_zscore_posterior *Per-statement factor-score posterior across factors*

Description

For a single statement, draws the posterior median z-score per factor with nested 50 percent (thick) and 95 percent (thin) credible-interval whiskers, stacked vertically. The x-axis is symmetric around zero so the zero reference is centred.

Usage

```
plot_zscore_posterior(fit, statement, ...)
```

Arguments

`fit`	A `bayesqm_fit`.
`statement`	Integer index or statement name.
`...`	Additional arguments forwarded to `plot()`.

Value

The input, invisibly.

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 `rstantools::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 (as percent strings).

`prior_summary.bayesqm_fit` *Prior summary for a bayesqm_fit*

Description

Returns the priors actually used when the model was fit, as a printable `bayesqm_prior` object. Method for `rstantools::prior_summary()`.

Usage

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

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

Arguments

`object`	A `bayesqm_fit`.
`...`	Unused.
`x`	A `bayesqm_prior` object.

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 companion 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 factors consistently across a bayesqm_fit*

Description

Replaces the default `f1..fK` factor labels everywhere they appear on the fit: posterior-mean and credible-interval loading matrices, the factor-score matrices, the factor-characteristics table, the correlation matrix, the flagged matrix, the distinguishing / consensus table column names, and the factor dimension of the raw `Lambda_draws` / `F_draws` arrays.

Usage

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

Arguments

`fit`	A `bayesqm_fit` object.
`new_names`	Character vector of length `K` with the new factor labels.

Value

The input fit with every factor label replaced.

Examples

```
fit <- demo_fit(N = 6, J = 10, K = 3, Td = 50, seed = 1)
fit <- rename_factors(fit, c("tradition", "innovation", "caution"))
colnames(fit$loa)
```

`run_bayes` *Fit the model across a range of K*

Description

Fits `fit_bayesian()` for K = 1..K_max and reports the ELPD peak (automated adoption), the Sivula (2025) parsimony diagnostic, and a case label (`agree`, `gap`, `reversed`) summarising their relationship. When the two agree the data supports that K; when they disagree the `gap` is itself a reportable finding.

Usage

```
run_bayes(
  Y,
  K_max = 5,
  stan_dir = NULL,
  elpd_diff_threshold = 4,
  se_ratio_threshold = 2,
  ...
)

select_k_peak(elpds, K_candidates)

select_k_sivula(
  elpds,
  loo_list,
  K_candidates,
  elpd_diff_threshold = 4,
  se_ratio_threshold = 2
)
```

Arguments

`Y`	A `qsort_data` object or `J x N` numeric matrix.
`K_max`	Largest K to try (default 5).
`stan_dir`	Optional override of the Stan model directory.
`elpd_diff_threshold`, `se_ratio_threshold`	Sivula rule thresholds (defaults 4 and 2).
`...`	Passed to `fit_bayesian()`.
`elpds`, `loo_list`, `K_candidates`	Internal inputs for `select_k_peak()` and `select_k_sivula()`.

Value

`run_bayes()` returns a `bayesqm_run` object carrying the list of fits, the ELPD comparison table, and the peak / Sivula / case verdict. `select_k_peak()` and `select_k_sivula()` return an integer K.

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

```
fit <- demo_fit(N = 6, J = 10, K = 2, Td = 50, seed = 1)
f <- file.path(tempdir(), "fig_loadings.pdf")
save_bayesqm_plot(f, plot_loading_posterior(fit))
unlink(f)
```

suggest_delta *Suggested separation* `delta` *from the forced distribution*

Description

Returns the standardized-scale width of one forced-distribution category, `1 / sd(forced positions)`. This is an alternative separation to the package default, the reliability-adjusted critical difference of `critical_delta()`.

Usage

```
suggest_delta(distribution)
```

Arguments

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

Value

A single numeric value: the standardized width of one forced-distribution category.

Examples

```
suggest_delta(c(2, 3, 4, 6, 8, 6, 4, 3, 2))
```

---

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