

Package: IJSE (via r-universe)

October 22, 2026

Type Package

Title Infinitesimal Jackknife Standard Errors for 'brms' Models

Version 0.3.4

Description Computes infinitesimal jackknife standard errors and covariance matrices for posterior expectations from 'brms' models or matched posterior draws and pointwise log likelihoods. Supports independent and clustered sampling units, transformed quantities, blockwise likelihood evaluation, and chain stability diagnostics. References: Ji et al. (2026) [doi:10.3102/10769986251379738](https://doi.org/10.3102/10769986251379738); Giordano and Broderick (2024) [doi:10.48550/arXiv.2305.06466](https://doi.org/10.48550/arXiv.2305.06466).

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports posterior (>= 1.4.0), stats

Suggests brms (>= 2.20.0), testthat (>= 3.0.0), knitr, rmarkdown, ggplot2 (>= 3.4.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 8.0.0

NeedsCompilation no

Author Feng Ji [aut, cre] (ORCID: <https://orcid.org/0000-0002-2051-5453>), JoonHo Lee [aut], Sophia Rabe-Hesketh [aut]

Maintainer Feng Ji <f.ji@utoronto.ca>

Repository <https://trtfj.r-universe.dev>

Date/Publication 2026-10-22 13:30:32 UTC

RemoteUrl <https://github.com/cran/IJSE>

RemoteRef HEAD

RemoteSha a01e430ca22f6293ea3de5c84751b9f82ca08051

Contents

IJSE-package	2
ijse	3
ijse_draws	5
summary.ijse	9
Index	12

IJSE-package	<i>IJ inference for posterior expectations</i>
--------------	--

Description

Compute infinitesimal jackknife sampling uncertainty from matched posterior draws and observation-level log likelihoods, with optional cluster aggregation.

Start with your available inputs

A fitted brms model	ijse returns structured inference.
Draws and log likelihoods	ijse_draws is backend independent.
Existing vector-only code	IJ_se preserves the original API.
A computed result	summary.ijse documents standard methods.

The matrix/callback interface requires no brms or Stan installation. The brms interface requires brms; installed synthetic examples can be used without compiling Stan or fitting a new model.

What the uncertainty means

The target is a posterior expectation, including a function evaluated jointly on every draw. Posterior draw covariance and IJ sampling covariance are different quantities. `vcov(result)` returns IJ covariance; `vcov(result, type = "posterior")` returns posterior covariance. `confint(result)` uses a normal reference and IJ SEs. IJ changes uncertainty, not estimator bias, and cluster IDs must reflect meaningful independent units.

Executable tutorials and calibration study

- `vignette("ijse", package = "IJSE")`: complete first calculation, covariance, transformations and callbacks without Stan.
- `vignette("ijse-brms", package = "IJSE")`: actual median-regression examples, explicit sampling IDs, joint contrasts and draw selection.
- `vignette("ijse-diagnostics", package = "IJSE")`: chain metadata, influence derivatives and reproduced input errors.

- `vignette("ijse-study", package = "IJSE")`: 40,000 exact-posterior datasets, 320 independent draw ensembles and 400 actual brms fits, with Monte Carlo uncertainty and explicit calibration limitations.

The three tutorial scripts are installed under `system.file("tutorials", package = "IJSE")`. The study vignette reads compact results; package checks do not rerun the MCMC study. Use `citation("IJSE")` for software and published-method citations.

Author(s)

Maintainer: Feng Ji <f.ji@utoronto.ca> ([ORCID](#))

Authors:

- Feng Ji <f.ji@utoronto.ca> ([ORCID](#))
- JoonHo Lee <jlee296@ua.edu>
- Sophia Rabe-Hesketh <sophiarh@berkeley.edu>

See Also

[ijse](#), [ijse_draws](#), [summary.ijse](#)

ijse

Infinitesimal jackknife inference for brms models

Description

Extract matched draws and likelihood contributions from a `brmsfit` object and compute IJ inference with [ijse_draws](#).

Usage

```
ijse(
  fit,
  cluster_var = NULL,
  variables = NULL,
  transform = NULL,
  draw_ids = NULL,
  block_size = NULL,
  save_influence = FALSE,
  diagnostics = c("all", "posterior", "none")
)

IJ_se(fit, cluster_var = NULL)
```

Arguments

<code>fit</code>	A fitted <code>brmsfit</code> object containing posterior draws.
<code>cluster_var</code>	Optional cluster IDs in the order of the likelihood columns returned by <code>brms::log_lik(fit)</code> . IDs must refer to the observations actually used in fitting, after any omissions or subsetting. IDs are not inferred from the formula. Save them separately: <code>brms</code> can drop unused data columns, so an absent <code>fit\$data\$group</code> is not a safe source of IDs.
<code>variables</code>	Optional exact parameter names. Defaults to parameters starting with <code>b_</code> , preserving <code>IJ_se()</code> behavior.
<code>transform</code>	Optional function applied to the selected draws; see ijse_draws . Select every required input with <code>variables</code> .
<code>draw_ids</code>	Optional strictly increasing indices of post-warmup draws, ordered by iteration within chain, then chain. The same indices are passed to likelihood extraction. Each retained chain requires at least two draws when <code>diagnostics = "all"</code> .
<code>block_size</code>	NULL (default) for ordinary likelihood extraction, or a positive integer for pointwise streaming. Streaming currently supports univariate models and requires <code>brms</code> pointwise-likelihood support. Unsupported models fail explicitly; use NULL for ordinary extraction.
<code>save_influence</code>	Whether to retain empirical influence values.
<code>diagnostics</code>	One of "all", "posterior", or "none"; see ijse_draws . The legacy <code>IJ_se()</code> wrapper uses "none" to avoid computing discarded diagnostics.

Details

MCMC chain diagnostics are retained for the selected or transformed quantities. The likelihood uses the `brms` default of conditioning on fitted group effects. This extraction choice does not establish IJ validity for arbitrary hierarchical models. For clustered working-independence models, supply cluster IDs explicitly; they are never inferred from a model formula.

Streaming respects observation permutations in `brms` prediction metadata. The tested `brms` release does not support pointwise evaluation for regression ARMA models; use ordinary extraction for those models. Covariance ARMA models are rejected because `brms` returns conditional LOO densities instead of joint likelihood factors. Supply one joint likelihood per independent series to `ijse_draws()`. Discrete truncated models also require explicit matrix input because the backend lower-bound normalization is not validated. Multivariate streaming is rejected because response-specific orderings may differ; the default dense path uses `brms`'s combined likelihood.

Measurement-error predictors require saved latent draws, for example `brms::save_pars(latent = TRUE)` when fitting. Otherwise `brms` may resample prediction latents, which do not match the fitted posterior draws. Even with saved latents, the native likelihood conditions on them; it is not an observed-data likelihood integrating measurement error. Missing-response models may return non-finite likelihoods, which are rejected. Use explicitly derived observed-data likelihoods with `ijse_draws()` when appropriate.

Value

`ijse()` returns the structured `ijse` object documented under [ijse_draws](#). `IJ_se()` returns the named numeric vector of standard errors, as in previous releases.

Choosing an interface

Use `ijse()` to retain estimates, both covariance types, intervals, diagnostics and optional influence. `IJ_se()` returns only IJ SEs with the same pooled formula. Ordinary extraction is the default. A positive `block_size` requests brms pointwise likelihood evaluation; backend prediction/preparation objects may still be large. This differs from `ijse_draws()`, whose positive block size controls a supplied callback.

Parameter extraction and likelihood extraction share `draw_ids`. Do not manually reorder parameter rows or pass cluster IDs for the original unfiltered dataset. Keep fitting-row keys with the actual model data and check `clustered`, `n_obs`, `n_units`, and `unit_ids`. Correct lengths alone cannot identify mismatched scientific units.

See Also

[ijse_draws](#), [summary.ijse](#)

Examples

```
# Uses an installed synthetic fit: no Stan compilation or new fitting.
if (requireNamespace("brms", quietly = TRUE)) {
  examples <- readRDS(system.file("extdata", "tutorial",
    "median-examples.rds", package = "IJSE"))
  fit <- examples$clustered$fit
  ids <- examples$clustered$cluster
  result <- ijse(fit, cluster_var = ids, save_influence = TRUE)
  stopifnot(result$clustered, result$n_units == 30L)
  summary(result)
  vcov(result)
  vcov(result, type = "posterior")
  confint(result, parm = "b_x", level = 0.90)
  head(result$unit_ids)
  stopifnot(isTRUE(all.equal(IJ_se(fit, ids), result$se)))
  streamed <- ijse(fit, cluster_var = ids, block_size = 40,
    diagnostics = "none")
  stopifnot(isTRUE(all.equal(vcov(streamed), vcov(result))))
  prediction <- ijse(fit, cluster_var = ids,
    variables = c("b_Intercept", "b_x"),
    transform = function(d) {
      cbind(median_at_x1 = d[, "b_Intercept"] + d[, "b_x"])
    }, diagnostics = "posterior")
  coef(prediction)
}
# For fitting instructions, run vignette("ijse-brms", package = "IJSE").
```

`ijse_draws`

Infinitesimal jackknife inference from posterior draws

Description

Compute sampling covariance of posterior expectations from matched posterior draws and pointwise log likelihoods, with optional cluster aggregation.

Usage

```
ijse_draws(
  draws,
  log_lik,
  cluster_var = NULL,
  variables = NULL,
  transform = NULL,
  n_obs = NULL,
  block_size = 1000L,
  save_influence = FALSE,
  chain_id = NULL,
  diagnostics = c("all", "posterior", "none")
)
```

Arguments

<code>draws</code>	A numeric matrix or data frame (draws by parameters), or a posterior draws object. Rows must correspond exactly to likelihood rows.
<code>log_lik</code>	A finite numeric matrix (draws by observations), or a function accepting observation indices and returning such a matrix for those columns. A callback must preserve the requested column order and draw row order.
<code>cluster_var</code>	Optional atomic vector of cluster IDs, in likelihood-column order. Missing IDs, including used explicit NA factor levels, are rejected. At least two clusters are required.
<code>variables</code>	Optional character vector of exact parameter names to select before applying transform. By default all parameters are selected.
<code>transform</code>	Optional function of the selected plain numeric draw matrix. Return a numeric vector, matrix, or data frame with one row per input draw. Named columns are recommended. Each returned column defines a posterior expectation of interest. Names on a returned vector are draw-row labels, not a quantity name; they are preserved and checked against available input labels. Use a one-column matrix or data frame to name a single transformed quantity.
<code>n_obs</code>	Number of likelihood columns. Required for a likelihood callback.
<code>block_size</code>	Maximum number of likelihood columns requested per callback block. Already materialized matrices use the compiled dense covariance path.
<code>save_influence</code>	Whether to retain the empirical influence matrix (parameters by resampling units). Default FALSE.
<code>chain_id</code>	Optional atomic vector identifying the chain of each draw. Within each chain, rows must be in iteration order. Inferred for posterior draws objects unless supplied explicitly. Each chain must have two draws when computing chain-specific IJ SEs. Missing chain IDs, including used explicit NA factor levels, are rejected.
<code>diagnostics</code>	One of "all" (chain-specific IJ SEs and posterior diagnostics), "posterior" (R-hat and ESS only), or "none". None of these options changes the pooled IJ estimate.

Details

Inputs have S posterior-draw rows, P selected or transformed quantities, and N observation-level likelihood columns. Supply log likelihoods excluding the prior, rather than a log posterior or a single joint-likelihood column. At least two draws and two sampling units are required. `variables` is applied before transform.

For quantity vector $g(\theta)$, write $C_i = \text{Cov}_{\theta|y}(g(\theta), \ell_i(\theta))$. Its estimate uses sample covariance over draws with denominator $S - 1$. For independent observations, $K = N$ and each unit contribution is $C_u = C_i$. For clusters, $K = J$ and $C_u = \sum_{i \in u} C_i$. The returned sampling covariance is

$$\hat{V}_{IJ} = \frac{K}{K-1} \sum_{u=1}^K (\hat{C}_u - \bar{C})(\hat{C}_u - \bar{C})^\top.$$

Here $\bar{C}$ is the average unit contribution. This is the finite-sample convention used by Ji et al. (2026).

The target is $E\{g(\theta) \mid y\}$, which can differ from $g\{E(\theta \mid y)\}$. Transforms use jointly sampled quantities. Fixed-reference predictors are supported; extra sampling variation from estimating the reference distribution is not included. IJ covariance is sampling covariance of posterior expectations, not posterior draw covariance. It does not remove estimator bias. Validity depends on sampling units, likelihood factorization, and model regularity. Cluster aggregation alone does not validate arbitrary latent-effects models, few clusters, time series, informative cluster sampling, or multiway dependence.

Value

An `ijse` list containing:

`estimate` Named vector of P posterior draw means.

`posterior_sd`, `posterior_vcov` Named posterior sample SDs and a $P \times P$ posterior sample covariance matrix.

`se`, `vcov` Named IJ SEs and a $P \times P$ IJ sampling covariance matrix. Both covariance matrices have quantity `dimnames`.

`n_draws`, `n_obs`, `n_units` Counts S , N , and K . Without clustering, `n_units` == `n_obs`.

`clustered` Whether cluster IDs were supplied. Unique per-row IDs still set this flag to `TRUE`, with observation-equivalent covariance.

`unit_ids` Exact original cluster IDs, named by unique display labels, in first-appearance order; `NULL` for independent observations.

`influence` `NULL` unless requested; otherwise a $P \times K$ matrix equal to $K\hat{C}_u$, with quantity rows and unit-label columns. See [summary.ijse](#) for interpretation.

`diagnostics` List with `mode`, `chain_ids`, `chain_se` (chains by quantities), `chain_relative_range`, `rhat`, `ess_bulk`, and `ess_tail`. Uncomputed entries are `NULL`. Posterior metrics and relative ranges are named by quantity.

`call` The matched function call.

Alignment and labels

Explicit row labels are checked when available on both sides. Names of a vector returned by `transform` are treated as draw-row labels. Automatic data-frame row numbers are positional. With unlabeled or incorrectly relabeled inputs, equal dimensions cannot establish correct row alignment. Labeled callback blocks must also agree with one another, including when the original draws are unlabeled. The first labeled block establishes that cross-block reference; unlabeled rows still require the caller's care. A callback must preserve observation-column order; it cannot be inferred.

Cluster and chain IDs are matched in order of first appearance. Their display labels use `make.unique(as.character(ids))`; distinct numeric IDs can otherwise print alike. Exact original cluster IDs are in `unit_ids`, named by those labels. Exact chain IDs are in `diagnostics$chain_ids` when diagnostics are enabled. Do not parse display labels to reconstruct IDs.

Diagnostics and memory

"all" computes chain-specific IJ SEs when each chain has at least two draws. R-hat and bulk/tail ESS require equally long chains with at least four draws each. The metrics describe selected or transformed quantities, not the precision of all quantity-likelihood covariances. Missing calculations are NULL; computed metrics can be NA, for example for constants. The chain relative range is $(\max(\text{SE}) - \min(\text{SE})) / \text{pooled_SE}$; it is a stability check, not a calibrated IJ Monte Carlo SE or a universal pass rule. For actual MCMC fits, also inspect sampler and nuisance-parameter diagnostics.

`block_size` limits callback requests, not storage of a matrix already supplied. Dense matrices use `stats::cov()`. Saved influence uses $P \times K$ storage; it is off by default. Clustered computations retain aggregated quantity-by-cluster contributions even when influence is not saved. Finite inputs can overflow covariance calculations; rescale when requested.

References

Ji, F., Lee, J., and Rabe-Hesketh, S. (2026). Valid Standard Errors for Bayesian Quantile Regression With Clustered and Independent Data. *Journal of Educational and Behavioral Statistics*, 51(5). doi:10.3102/10769986251379738. Giordano, R. and Broderick, T. (2024). The Bayesian Infinitesimal Jackknife for Variance. doi:10.48550/arXiv.2305.06466.

See Also

[ijse](#), [summary.ijse](#), [IJSE-package](#). Run `vignette("ijse", package = "IJSE")` for a complete executable introduction.

Examples

```
set.seed(12)
y <- c(-1, 0, 0.5, 1, 2)
v <- 1 / (length(y) + 1 / 4)
mu <- rnorm(2000, v * sum(y), sqrt(v))
draws <- cbind(mu = mu)
ll <- vapply(y, function(yi) dnorm(yi, mu, log = TRUE),
             numeric(length(mu)))
result <- ijse_draws(draws, ll)
summary(result)
```

```
vcov(result)
confint(result)
ijse_draws(draws, ll,
  transform = function(x) cbind(mu_squared = x[, 1]^2))
# A callback avoids storing the whole likelihood matrix.
ijse_draws(draws, function(i) ll[, i, drop = FALSE],
  n_obs = length(y), block_size = 2)
```

summary.ijse

Summarize IJ inference

Description

Standard methods for posterior expectations and their IJ sampling covariance. Normal intervals are centered on posterior expectations and use IJ SEs; they are not posterior credible intervals or a small-cluster correction.

Usage

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

## S3 method for class 'ijse'
vcov(object, ..., type = c("ij", "posterior"))

## S3 method for class 'ijse'
influence(model, ...)

## S3 method for class 'ijse'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'ijse'
summary(object, level = 0.95, ...)

## S3 method for class 'ijse'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.ijse'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

object, x, model	An ijse object, or its summary for the summary print method.
...	Reserved for future arguments; must be empty.
type	For <code>vcov()</code> , select IJ sampling covariance (default) or covariance of the posterior draws with <code>type = "posterior"</code> .
parm	Parameter names or integer indices; defaults to all quantities.

level	Confidence level strictly between zero and one.
digits	Number of significant digits used for printing.

Details

`confint()` uses the posterior expectation plus or minus a standard-normal critical value times the IJ SE. It is symmetric and uses a normal reference, even for clustered results. It is not an equal-tailed posterior credible interval, an IJ Monte Carlo interval, or a t correction. The upper-tail calculation avoids rounding a valid probability to one when `level` is very close to one. Below `level = 1e-8`, the central inverse-probability expansion $\sqrt{\pi/2} \text{level}$ avoids losing a tiny positive level; its omitted relative correction is below double precision. Floating-point precision can limit display of extreme confidence levels and very narrow interval endpoints.

`SE_ratio` in the summary is `IJ_SE / Posterior_SD`; it is NA when the posterior SD is zero. A larger ratio alone does not prove calibrated inference. The two `vcov()` choices estimate distinct objects. Posterior covariance is unavailable in objects saved before version 0.3.0; recompute those objects to request it.

Saved `influence()` values are $K\hat{C}_u$, not zero-centered influence values. Divide by `n_units` to obtain case-weight derivative estimates. For a small likelihood-weight change Δw_u , $\Delta E(g | y) \approx \hat{C}_u \Delta w_u$; downweighting reverses the derivative's sign. Deleting a whole unit is a finite perturbation and need not equal this local approximation. The rows need not have mean zero. Sampling covariance centers across units, so `cov(t(influence(result))) / result$n_units` reproduces `vcov(result)`. Use `result$unit_ids` for exact cluster IDs; columns use unique display labels. Independent observations use their likelihood-column indices.

In a finite-draw importance-reweighting check, normalized empirical weights differentiate with denominator S , whereas IJSE uses sample covariance with denominator $S-1$. Multiply that numerical derivative by $S/(S-1)$ when checking equality. This is a testing identity, not an extra correction to apply to the reported IJ SE.

Value

`coef()` returns a named vector of posterior expectations. `vcov()` returns the $P \times P$ IJ sampling covariance by default, or posterior draw covariance for `type = "posterior"`. `confint()` returns a quantities-by-two matrix, in the requested parm order, with percentage-labeled columns. When the percentage labels would coincide at display precision, columns are named lower and upper. `influence()` returns the saved quantities-by-units matrix or errors when influence was not retained. `summary()` returns a `summary.ijse` list: coefficients (Estimate, Posterior_SD, IJ_SE, SE_ratio, lower and upper limits), counts, clustered, `unit_ids`, `level`, diagnostics and call. Print methods invisibly return their input.

See Also

[ijse_draws](#), [ijse](#).

Examples

```
set.seed(1)
y <- c(-1, 0, 0.5, 1, 2)
v <- 1 / (length(y) + 1 / 4)
```

```
d <- cbind(theta = rnorm(1000, v * sum(y), sqrt(v)))
ll <- vapply(y, function(yi) dnorm(yi, d[, 1], log = TRUE),
            numeric(nrow(d)))
result <- ijse_draws(d, ll, save_influence = TRUE)
coef(result)
vcov(result)
confint(result, level = 0.90)
summary(result)
vcov(result, type = "posterior")
influence(result) / result$n_units
stopifnot(isTRUE(all.equal(
  cov(t(influence(result))) / result$n_units, vcov(result))))
```

Index

`coef.ijse(summary.ijse)`, [9](#)
`confint.ijse(summary.ijse)`, [9](#)

`IJ_se`, [2](#)
`IJ_se(ijse)`, [3](#)
`IJSE (IJSE-package)`, [2](#)
`ijse`, [2](#), [3](#), [3](#), [8](#), [10](#)
`IJSE-package`, [2](#)
`ijse_draws`, [2–4](#), [5](#), [5](#), [10](#)
`influence.ijse(summary.ijse)`, [9](#)

`print.ijse(summary.ijse)`, [9](#)
`print.summary.ijse(summary.ijse)`, [9](#)

`summary.ijse`, [2](#), [3](#), [5](#), [7](#), [8](#), [9](#)

`vcov.ijse(summary.ijse)`, [9](#)