Package ‘flexstanr’

August 25, 2026

Title Portable Backend Layer for 'Stan' Models

Version 0.2.0

Description Gives a 'Stan'-based R package one interface for fitting its models through either 'rstan' or 'cmdstanr', neither of which is required to install this package (install whichever you use). Collects and validates sampler options, guarding against mixing one backend's argument vocabulary into the other, dispatches the fit to the chosen backend, and exposes backend-agnostic accessors for reading posterior draws, extracting parameters, and running generated quantities. The host package supplies its own compiled models; flexstanr resolves them from the calling package at run time, so the same code works whichever backend is installed.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports methods, parallelly, tools

Suggests cmdstanr, desc, knitr, pkgload, posterior, rmarkdown, roxygen2, rstan (>= 2.18.1), spelling, testthat (>= 3.2.0), withr

VignetteBuilder knitr

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

Config/testthat/edition 3

URL https://accidda.github.io/flexstanr/, https://github.com/ACCIDDA/flexstanr

BugReports https://github.com/ACCIDDA/flexstanr/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Carl Pearson [aut, cre] (ORCID: <https://orcid.org/0000-0003-0701-7860>), Weston Voglesonger [aut]

Maintainer Carl Pearson <carl.ab.pearson@gmail.com>

Repository CRAN

Date/Publication 2026-08-25 21:30:02 UTC

Contents

`backend_draws_array` *Posterior draws of a fit as an iterations x chains x parameters array*

Description

Posterior draws of a fit as an iterations x chains x parameters array

Usage

```
backend_draws_array(raw_fit)
```

Arguments

`raw_fit`	a backend-native fit object (an rstan `stanfit` or a cmdstanr `CmdStanMCMC`).

Value

a 3-D array, dimensions iterations x chains x parameters.

Examples

```
## Not run:
draws <- backend_draws_array(fit)
dim(draws) # iterations x chains x parameters

## End(Not run)
```

backend_extract *Extract parameters from a fit, in a chosen format*

Description

The backend-agnostic way to read a fit's posterior. `format` selects the representation, and each format has the same shape whichever backend produced the fit, so downstream math does not have to branch on the backend:

- `"list"` (the default) matches `rstan::extract()`: a named list with one entry per parameter, chains merged into a single draw dimension that comes first, a true scalar parameter collapsed to a bare length-`S` vector, and a `vector[1]` kept as an `S x 1` matrix. The rstan backend delegates to `rstan::extract()` itself; the cmdstanr backend reshapes its draws to match.
- `"draws"` returns a posterior::draws_array (iteration x chain x variable), keeping the chain structure and the flat Stan variable names.
- `"matrix"` returns a plain base matrix with one row per draw (chains stacked) and one column per flat variable, the shape `backend_generate_quantities()` takes as `draws_mat`.

`"draws"` and `"matrix"` preserve iteration-chain draw order on both backends. `"list"` does not: `rstan::extract()` permutes draws by default while the cmdstanr path does not, which is immaterial for the exchangeable -sample uses these draws are put to but means the two backends' `"list"` output agrees as a sample, not draw for draw.

Usage

```
backend_extract(
  raw_fit,
  pars = NULL,
  format = c("list", "draws", "matrix"),
  ...
)
```

Arguments

`raw_fit`	a backend-native fit object (an rstan `stanfit` or a cmdstanr `CmdStanMCMC`).
`pars`	character vector of parameter names to extract (a single name is fine). Use the base name of a container parameter (`"theta"`, not `"theta[1]"`). `NULL`, the default, extracts every parameter, including `lp__`.
`format`	the representation to return, one of `"list"` (the default), `"draws"`, or `"matrix"`; see Description.
`...`	forwarded verbatim to the backend's own extractor, and accepted only by `format = "list"`. Arguments that change the return shape (for instance `rstan::extract()`'s `permuted = FALSE`) take the result outside the contract above; prefer `format = "draws"` for a chain-preserving array.

Value

the fit's draws for `pars`, in the requested `format`.

See Also

`backend_draws_array()` for the raw iterations x chains x parameters array, and `backend_generate_quantities()`, whose `draws_mat` argument takes `format = "matrix"` output.

Examples

```
## Not run:
# rstan::extract()-compatible, the shape most existing code expects
post <- backend_extract(fit, pars = c("beta", "sigma"))
colMeans(post$beta)

# every parameter, no `pars` needed
all_post <- backend_extract(fit)

# backend-neutral posterior draws, chains kept
draws <- backend_extract(fit, format = "draws")

# a draws x parameters matrix, ready for backend_generate_quantities()
mat <- backend_extract(fit, format = "matrix")

## End(Not run)
```

---

backend_generate_quantities

Run generated quantities against a fit and return a parameter matrix

---

Description

Run generated quantities against a fit and return a parameter matrix

Usage

```
backend_generate_quantities(
  raw_fit,
  data,
  draws_mat,
  pars,
  model_name = NULL,
  package = NULL
)
```

Arguments

raw_fit	a backend-native fit object (an rstan `stanfit` or a cmdstanr `CmdStanMCMC`).
data	the Stan data list for the generated-quantities run.
draws_mat	a draws matrix (rows = draws, columns = parameters), as returned by `backend_extract(raw_fit, format = "matrix")`. Used by the rstan backend; the cmdstanr backend runs generated quantities against the fit's own draws and ignores this argument.
pars	name of the generated parameter to return.
model_name	name of the model whose generated-quantities block to run. Required by the cmdstanr backend, which recompiles the model to run it; ignored by rstan, which reuses the model carried on `raw_fit`.
package	the host package the model belongs to; defaults to the calling package (see `fit_model()`). Only used by the cmdstanr backend.

Value

a matrix of the requested generated parameter (rows = draws).

Examples

```
## Not run:
gen <- backend_generate_quantities(fit, data = data_list,
                                   draws_mat = as.matrix(fit), pars = "y_rep")

## End(Not run)
```

---

backend_has_draws *Does a fit object carry usable posterior draws?*

Description

Detect the degenerate "no draws" case after a fit, so a caller can fail loudly instead of returning an empty fit. This is backend-aware: rstan returns a mode-2 `stanfit` with an empty `@sim` when the sampler fails to initialize (rather than erroring), while cmdstanr exposes its draws through `$draws()`. Unrecognized objects (e.g. test mocks) are treated as having draws so they pass through untouched.

Usage

```
backend_has_draws(raw_fit)
```

Arguments

raw_fit	a backend-native fit object (an rstan `stanfit` or a cmdstanr `CmdStanMCMC`).

Value

logical; TRUE if the fit carries usable draws.

Examples

```
# Unrecognized objects are treated as carrying draws (pass-through).
backend_has_draws(list())
```

`fit_model` *Fit a Stan model through the chosen backend*

Description

Dispatches a fit to the backend recorded on `stan_opts` (from `stan_options()`). The compiled model is resolved by `model_name` from the calling package: for `"rstan"`, `package::stanmodels[[model_name]]`; for `"cmdstanr"`, `inst/stan/<model_name>.stan` under `package`. The calling package is detected automatically and can be overridden with `package`.

Usage

```
fit_model(
  model_name,
  dat_stan,
  init,
  stan_opts,
  drop_pars = NULL,
  package = NULL
)
```

Arguments

`model_name`	name of the Stan model; used to look up the compiled model in the calling package's `stanmodels` (rstan) and to locate the `.stan` source file under its `inst/stan/` (cmdstanr).
`dat_stan`	the Stan data list.
`init`	the init list, sized to the chain count.
`stan_opts`	the validated `stan_options()` list (carrying a `backend` element).
`drop_pars`	character vector of parameter names to exclude from the saved draws, or `NULL` to keep everything. Honored by rstan; cmdstanr cannot drop parameters and warns if any are requested.
`package`	name of the host package whose model is being fit. Defaults to the package that called `fit_model()`, which is correct for the usual case of a host package fitting one of its own models.

Value

the backend's fit object (a `stanfit` or `CmdStanMCMC`).

Examples

```
## Not run:
# From inside a host package that ships a compiled `coverage` model:
opts <- stan_options(chains = 2, iter = 500, seed = 1)
fit <- fit_model("coverage", dat_stan = data_list, init = init_list,
                 stan_opts = opts)

## End(Not run)
```

`stan_options` *Stan Sampler Options*

Description

Collects sampler arguments for the chosen backend, validating common arguments and forwarding all other same-backend arguments **verbatim** to the native sampler. The native sampler remains responsible for validating those forwarded arguments. Mixing one backend's known vocabulary into the other errors with a hint. The model object is supplied separately (via `fit_model()`), while `data` and `init` are constructed internally, so none of these may be set here. `chains` defaults to 4 so downstream code can always size per-chain structures from it.

Usage

```
stan_options(
  ...,
  chains = 4L,
  backend = "rstan",
  threading = FALSE,
  max_cores = NULL
)
```

Arguments

`...`	arbitrary sampler arguments forwarded verbatim to the chosen backend's sampler. Use the backend's own names: for `"rstan"`, the `rstan::sampling()` arguments (`iter`, `cores`, `seed`); for `"cmdstanr"`, the `$sample()` arguments (`iter_warmup`, `iter_sampling`, `parallel_chains`, ...).
`chains`	number of Markov chains to run. Defaults to 4 for both backends.
`backend`	which Stan interface to target, one of `"rstan"` (default) or `"cmdstanr"`. Determines which argument vocabulary is accepted and which sampler `fit_model()` calls. Both backends are optional; selecting one errors if its package is not installed.

threading — TRUE to let flexstanr use the machine's spare cores: it splits the cores the process is allowed to use across the chains and the within-chain (reduce_sum) threads, and messages what it chose. FALSE (the default) leaves parallelism untouched, so you can still set cores (rstan) or parallel_chains / threads_per_chain (cmdstanr) by hand. A model that cannot use the offered threads should say so; see test_threaded().

max_cores — when threading = TRUE, an optional cap on the cores used. NULL (the default) uses all available cores; set it to leave some free / cap usage for other work. Ignored when threading = FALSE.

Value

a named list of validated sampler arguments, carrying a backend element recording the backend it was built for

See Also

test_threaded()

Examples

```
if (requireNamespace("rstan", quietly = TRUE)) {
  stan_options()
  stan_options(chains = 2, iter = 500)
  stan_options(chains = 4, threading = TRUE)  # allocate spare cores to threads
}
if (requireNamespace("cmdstanr", quietly = TRUE)) {
  stan_options(backend = "cmdstanr", parallel_chains = 4, iter_warmup = 500)
}
```

---

test_threaded *Does a set of sampler options ask for within-chain threading?*

Description

A host package's fit function calls this to learn whether the caller requested within-chain threading – via stan_options(threading = TRUE), or by setting threads_per_chain directly – so it can warn when its model cannot make use of the offered threads (for example a model with no reduce_sum term, or one the host did not compile for threading). It reports what the *options* ask for, not whether the model can honor it: only the model's author knows that, which is why the check (and any resulting warning) belongs in the host's fit function rather than in flexstanr.

Usage

```
test_threaded(stan_opts)
```

Arguments

stan_opts	a stan_options() result.

Value

logical; TRUE if the options request more than one thread per chain, otherwise FALSE.

Examples

```
if (requireNamespace("rstan", quietly = TRUE)) {
  test_threaded(stan_options(chains = 2))              # FALSE (not requested)
}
test_threaded(list(threads_per_chain = 4L))            # TRUE
```

---

use_flexstanr *Wire flexstanr into a host package*

Description

A one-time setup helper, in the spirit of usethis's usethis::use_package(), that declares flexstanr as a dependency of the host package you run it from. It adds flexstanr to the host's Imports, optionally records a Remotes entry for a non-CRAN install, and writes a generated re-export file (R/flexstanr.R) so host::stan_options() keeps resolving and the host's internal calls to fit_model() / the backend_* accessors are imported. It does not add a Stan backend: flexstanr requires neither rstan nor cmdstanr, so the host declares whichever backend it uses.

The re-export file is generated: it carries a do-not-edit banner and is overwritten on each run, so re-run use_flexstanr() to pick up changes to flexstanr's re-exported surface. flexstanr resolves the host's own compiled models automatically from the calling package.

Usage

```
use_flexstanr(
  path = ".",
  min_version = NULL,
  remote = NULL,
  type = "Imports",
  reexport = TRUE,
  reexport_file = file.path("R", "flexstanr.R")
)
```

Arguments

path	path to the host package's root (the directory containing its DESCRIPTION). Defaults to the working directory.

min_version	minimum flexstanr version for the Imports entry. NULL (the default) pins to the currently-installed flexstanr version; pass a version string to pin explicitly, or FALSE for no constraint.
remote	optional owner/repo (or any remotes-style spec) recorded as a Remotes entry so remotes / pak can install flexstanr off-CRAN. NULL (the default) records nothing, which is what you want for the CRAN package; pass e.g. "ACCIDDA/flexstanr" to install a development build.
type	the DESCRIPTION field flexstanr is added to. "Imports" by default, as with usethis::use_package().
reexport	whether to (over)write the generated R/flexstanr.R re-export file. TRUE by default; set FALSE to only edit DESCRIPTION.
reexport_file	path, relative to path, of the generated re-export file.

Value

the host package path, invisibly.

Examples

```
## Not run:
# from the root of your Stan package (CRAN flexstanr, pinned to installed):
flexstanr::use_flexstanr()

# track a development build off GitHub instead:
flexstanr::use_flexstanr(remote = "ACCIDDA/flexstanr")

## End(Not run)
```

Index