Package: stanflow (via r-universe)

July 23, 2026

Title A Mildly Opinionated Stan Bayesian Workflow

Version 0.1.0.9000

Date 2026-01-24

Description Streamlines access to the Stan ecosystem by attaching core workflow packages, surfacing conflicts, and providing helpers to set up Stan interfaces with sensible defaults. Facilitates the Bayesian Workflow as described in Gelman et al. (2020) <https://arxiv.org/abs/2011.01808>.

License BSD_3_clause + file LICENSE

URL https://stanflow.visruth.com/

BugReports https://github.com/VisruthSK/stanflow/issues/

Depends R (>= 4.2.0)

Imports bayesplot, cli, fastmatch, loo, posterior, projpred, shinystan, withr

Suggests brms, cmdstanr, knitr, quarto, rstan, rstanarm, testthat (>= 3.0.0)

VignetteBuilder quarto

Additional_repositories https://community.r-multiverse.org https://stan-dev.r-universe.dev

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev zlib1g-dev

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

Date/Publication 2026-06-28 05:39:14 UTC

RemoteUrl https://github.com/VisruthSK/stanflow

RemoteRef HEAD

RemoteSha 6abe3e4a5bb56b6b454b563a3c6f91fd6a34b0df

Contents

`flow_check` *Print stanflow status and conflicts*

Description

Print a consolidated status report showing attached packages, available interfaces, and any conflicts.

Usage

```
flow_check(check_updates = FALSE, only = NULL)
```

Arguments

`check_updates`	If `TRUE`, checks for stable updates to stanflow packages.
`only`	Set this to a character vector to restrict to conflicts only between the provided packages and loaded stanflow packages.

Value

Invisibly returns the character vector that was printed.

Examples

```
flow_check()
```

scan_usage *Find used functions and packages*

Description

This function is primarily exported for developers. The scanner itself is generic and requires an explicit package universe; `stan_cite()` is the Stan-configured entry point. The scanning is wholly static (AST parsing), so there are a number of restrictions on what calls are recognized: calls to `library()`, `require()`, `requireNamespace()`, or `use()` are all recognized as attaching a package.

Usage

```
scan_usage(
  path = ".",
  allowed_packages,
  export_index,
  origin_map,
  ignore_unqualified_functions = .stdlib_funs,
  strict = FALSE,
  skip_dirs = .scan_skip_dirs,
  metapackages = NULL,
  use_knitr = FALSE,
  quiet = getOption("stanflow.quiet", FALSE)
)
```

Arguments

`path`	A single project directory (searched recursively) or a vector of files (.R/.Rmd/.qmd).
`allowed_packages`	Character vector of package namespaces to attribute calls to.
`export_index`	Named list mapping function names to packages.
`origin_map`	Named character vector mapping `pkg::fun` keys to the origin package.
`ignore_unqualified_functions`	Defaults to exports from base R packages listed in `stdlib_funs()`. Character vector of function names to ignore when attributing (unqualified) calls to Stan packages. Calls like `rstan::plot()` will NOT be ignored even if `plot` is in `ignore_unqualified_functions`, since they are namespaced.
`strict`	If `TRUE` (default), only count unqualified function calls whose origin can be determined exactly from the static scan, including attachment-order tie-breaks when the winner is unambiguous from the file. Unresolved calls are warned about and omitted.
`skip_dirs`	Character vector of directory names to skip when scanning a directory. Defaults to `.scan_skip_dirs`.
`metapackages`	Named list mapping attached package names to additional packages that should be treated as co-attached for unqualified resolution. Defaults to `NULL`.

use_knitr	Logical. If TRUE, parse .Rmd and .qmd files with knitr::purl(). This is more accurate for knitr/quarto chunk extraction but much slower than the default in-house parser. Defaults to FALSE.
quiet	Logical. If TRUE, suppresses status messages.

Details

Explicit package references from library(), require(), requireNamespace(), use(), and pkg::fun are only recorded when their package is included in allowed_packages. Unqualified function calls are only attributed when a package is attached via library() or require() in the same file and allowed_packages, export_index, and origin_map describe how to resolve them. Attaching a metapackage can also be treated as attaching additional packages when metapackages is supplied. When multiple attached packages export the same unqualified function, attachment order is respected: the most recently attached matching package whose attachment appears before the call is treated as the winner. Known reexports are remapped to their origin packages; missing mappings fall back to the resolved package.

Value

A list of packages, resolved functions, and ambiguous function calls.

Examples

```
path <- tempfile(fileext = ".R")
writeLines(
  c(
    "# one messy analysis file",
    "library(stats)",
    "requireNamespace(\"utils\")",
    "filter(1:10, rep(1, 3))",
    "utils::head(letters)"
  ),
  path
)
scan_usage(
  path,
  allowed_packages = c("stats", "utils"),
  export_index = list(filter = "stats"),
  origin_map = c("stats::filter" = "stats"),
  ignore_unqualified_functions = character(),
  quiet = TRUE
)
unlink(path)
```

---

setup_brms *Setup brms*

Description

Configures brms to use available cores and sets the backend. Prefer setup_interface() for user-facing setup since it performs argument validation and defaults; setup_brms() assumes inputs are already checked.

Usage

```
setup_brms(quiet, brms_backend, cores, dry_run = FALSE)
```

Arguments

quiet	Logical. If TRUE, suppresses status messages. This cannot suppress cmdstan messages.
brms_backend	Character. The brms backend to use. Defaults to getOption("brms.backend", "cmdstanr") and must be one of c("cmdstanr", "rstan").
cores	Integer. Number of cores to use. Defaults to getOption("mc.cores"). You must set options(mc.cores = ...) or pass cores explicitly.
dry_run	Logical. If TRUE, previews mutating setup actions without installing, attaching, changing options, or prompting. Dry-run output is shown even when quiet = TRUE.

Value

Returns NULL invisibly.

Examples

```
## Not run:
setup_brms(quiet = TRUE, brms_backend = "cmdstanr", cores = 2)

## End(Not run)
```

---

setup_cmdstanr *Setup cmdstanr and CmdStan*

Description

Checks the C++ toolchain, locates CmdStan, and installs or upgrades CmdStan if needed. Prefer setup_interface() for user-facing setup since it performs argument validation and defaults; setup_cmdstanr() assumes inputs are already checked.

Usage

```
setup_cmdstanr(
  quiet,
  force,
  reinstall = FALSE,
  check_updates = FALSE,
  cores,
  dry_run = FALSE
)
```

Arguments

quiet	Logical. If TRUE, suppresses status messages. This cannot suppress cmdstan messages.
force	Logical. If TRUE, allows installation in non-interactive sessions.
reinstall	Logical. If TRUE, forces re-installation.
check_updates	Logical. If TRUE, checks for CmdStan updates.
cores	Integer. Number of cores to use. Defaults to getOption("mc.cores"). You must set options(mc.cores = ...) or pass cores explicitly.
dry_run	Logical. If TRUE, previews mutating setup actions without installing, attaching, changing options, or prompting. Dry-run output is shown even when quiet = TRUE.

Value

Returns TRUE invisibly when no install/upgrade is needed. Otherwise, returns NULL invisibly after installation.

Examples

```
## Not run:
setup_cmdstanr(
  quiet = TRUE,
  force = TRUE,
  reinstall = FALSE,
  check_updates = FALSE,
  cores = 2
)

## End(Not run)
```

setup_interface *Setup and Load Stan Interfaces*

Description

This function ensures specific Stan interfaces are installed, configured, and loaded. It handles package installation (from R-multiverse/CRAN (stable) or Stan universe (dev)) and performs necessary one-time setup (like installing CmdStan).

Usage

```
setup_interface(
  interface = c("brms", "cmdstanr", "rstan", "rstanarm"),
  cores = getOption("mc.cores"),
  quiet = getOption("stanflow.quiet", FALSE),
  force = FALSE,
  reinstall = FALSE,
  check_updates = FALSE,
  dev = FALSE,
  brms_backend = c("cmdstanr", "rstan"),
  rstan_auto_write = TRUE,
  dry_run = FALSE
)
```

Arguments

interface	A character vector. Select at least one of: "brms", "cmdstanr", "rstan", "rstanarm".
cores	Integer. Number of cores to use. Defaults to `getOption("mc.cores")`. You must set `options(mc.cores = ...)` or pass `cores` explicitly.
quiet	Logical. If `TRUE`, suppresses status messages. This cannot suppress cmdstan messages.
force	Logical. If `TRUE`, allows installation in non-interactive sessions.
reinstall	Logical. If `TRUE`, forces re-installation.
check_updates	Logical. If `TRUE`, checks for CmdStan updates.
dev	Logical. If `FALSE` (default), installs stable releases from R-multiverse or CRAN. If `TRUE`, installs development versions from Stan R-universe.
brms_backend	Character. The `brms` backend to use. Defaults to `getOption("brms.backend", "cmdstanr")` and must be one of `c("cmdstanr", "rstan")`.
rstan_auto_write	Logical. If `TRUE` (default), sets `rstan::rstan_options(auto_write = TRUE)`
dry_run	Logical. If `TRUE`, previews mutating setup actions without installing, attaching, changing options, or prompting. Dry-run output is shown even when `quiet = TRUE`.

Details

The setup functions are exported (e.g., `setup_brms()`) for transparency. Each function has some side effects, mainly setting `mc.cores`, see the function for specifics.

Value

Returns attached package names invisibly. With `dry_run = TRUE`, returns the package names that would be attached.

Examples

```
## Not run:
options(mc.cores = 2)
setup_interface("cmdstanr", quiet = TRUE)
setup_interface(
  c("brms", "cmdstanr"),
  brms_backend = "cmdstanr",
  quiet = TRUE
)

## End(Not run)
```

`setup_rstan` *Setup rstan*

Description

Configures `rstan` to use available cores and write compiled models to disk. Prefer `setup_interface()` for user-facing setup since it performs argument validation and defaults; `setup_rstan()` assumes inputs are already checked.

Usage

```
setup_rstan(quiet, cores, rstan_auto_write, dry_run = FALSE)
```

Arguments

`quiet`	Logical. If `TRUE`, suppresses status messages. This cannot suppress cmdstan messages.
`cores`	Integer. Number of cores to use. Defaults to `getOption("mc.cores")`. You must set `options(mc.cores = ...)` or pass `cores` explicitly.
`rstan_auto_write`	Logical. If `TRUE` (default), sets `rstan::rstan_options(auto_write = TRUE)`
`dry_run`	Logical. If `TRUE`, previews mutating setup actions without installing, attaching, changing options, or prompting. Dry-run output is shown even when `quiet = TRUE`.

Value

Returns NULL invisibly.

Examples

```
## Not run:
setup_rstan(quiet = TRUE, cores = 2, rstan_auto_write = TRUE)

## End(Not run)
```

`setup_rstanarm` *Setup rstanarm*

Description

Configures `rstanarm` to use available cores. Prefer `setup_interface()` for user-facing setup since it performs argument validation and defaults; `setup_rstanarm()` assumes inputs are already checked.

Usage

```
setup_rstanarm(quiet, cores, dry_run = FALSE)
```

Arguments

`quiet`	Logical. If `TRUE`, suppresses status messages. This cannot suppress cmdstan messages.
`cores`	Integer. Number of cores to use. Defaults to `getOption("mc.cores")`. You must set `options(mc.cores = ...)` or pass `cores` explicitly.
`dry_run`	Logical. If `TRUE`, previews mutating setup actions without installing, attaching, changing options, or prompting. Dry-run output is shown even when `quiet = TRUE`.

Value

Returns NULL invisibly.

Examples

```
## Not run:
setup_rstanarm(quiet = TRUE, cores = 2)

## End(Not run)
```

stan_cite *Cite Stan packages in a project/files*

Description

stan_cite() generates the correct citations for Stan packages in a directory or set of files. Quarto (.qmd) and R Markdown (.Rmd) documents are scanned by extracting R code chunks directly from the source text by default. Setting use_knitr = TRUE switches to knitr::purl(), which is more accurate for knitr/quarto chunk handling but much slower. stan_cite() uses some simple heuristics to guess which packages export functions, and also attempts to map re-exports to their origin package. Calls to library(), require(), requireNamespace(), or use() are all recognized as attaching a package.

Usage

```
stan_cite(
  path = ".",
  strict = TRUE,
  format = c("bibtex", "bibentry"),
  skip_dirs = .scan_skip_dirs,
  ignore_unqualified_functions = .stdlib_funs,
  use_knitr = FALSE,
  quiet = getOption("stanflow.quiet", FALSE)
)
```

Arguments

path	A single project directory (searched recursively) or a vector of files (.R/.Rmd/.qmd).
strict	If TRUE (default), only count unqualified function calls whose origin can be determined exactly from the static scan, including attachment-order tie-breaks when the winner is unambiguous from the file. Unresolved calls are warned about and omitted.
format	One of "bibtex" or "bibentry", specifying the return format.
skip_dirs	Defaults to directories listed in scan_skip_dirs. Character vector of directory names to skip when scanning a directory.
ignore_unqualified_functions	Defaults to exports from base R packages listed in stdlib_funs(). Character vector of function names to ignore when attributing (unqualified) calls to Stan packages. Calls like rstan::plot() will NOT be ignored even if plot is in ignore_unqualified_functions, since they are namespaced.
use_knitr	Logical. If TRUE, parse .Rmd and .qmd files with knitr::purl(). This is more accurate for knitr/quarto chunk extraction but much slower than the default in-house parser. Defaults to FALSE.
quiet	Logical. If TRUE, suppresses status messages.

Details

The parsing is handled by `scan_usage()`; `stan_cite()` owns the citation lookups.

Value

A BibTeX character vector or a bibentry object.

Examples

```
path <- tempfile(fileext = ".R")
writeLines(
  c(
    "# one messy analysis file",
    "library(posterior)",
    "requireNamespace(\"loo\")",
    "draws <- as_draws(list(mu = rnorm(10)))",
    "posterior::rhat(draws)",
    "loo::loo(matrix(1))"
  ),
  path
)

stan_cite(path, quiet = TRUE)
stan_cite(path, format = "bibentry", quiet = TRUE)
unlink(path)
```

`stan_logo` *Print the Stan ASCII art logo*

Description

Print the Stan ASCII art logo

Usage

```
stan_logo()
```

Value

Invisibly returns the character vector that was printed.

Examples

```
stan_logo()
```

stan_repos *Stan package repositories*

Description

Stan package repositories

Usage

```
stan_repos(dev = FALSE)
```

Arguments

dev	Include the development r-universe repo–don't use this unless you need the latest commits.

Value

Character vector of repository URLs.

Examples

```
stan_repos()
stan_repos(dev = TRUE)
```

stanflow_conflicts *Conflicts between stanflow and other packages*

Description

List conflicts between stanflow packages and other attached packages.

Usage

```
stanflow_conflicts(only = NULL)

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

Arguments

only	Defaults to NULL. Set this to a character vector to restrict to conflicts only between the provided packages and loaded stanflow packages.
x	A stanflow_conflicts object, usually from stanflow_conflicts().
...	Unused. Included for consistency with base::print().

Details

There are several conflicts that are deliberately ignored: `diag`, `drop`, `match`, `\%in\%`, `mad`, `sd`, and `var` from posterior.

Adapted from `tidyverse::tidyverse_conflicts()` for stanflow's package set.

Value

Invisibly returns `x`.

Examples

```
stanflow_conflicts()
stanflow_conflicts(c("base"))
conflicts <- stanflow_conflicts()
print(conflicts)
```

---

stanflow_deps *List all stanflow dependencies*

Description

Returns a data frame of Stan workflow packages and their local/remote versions. When `check_updates = FALSE`, remote versions are not queried and the `remote` and `behind` columns are `NA` and `FALSE`, respectively.

Adapted from `tidyverse::tidyverse_deps()`.

Usage

```
stanflow_deps(recursive = FALSE, dev = FALSE, check_updates = TRUE)
```

Arguments

`recursive`	If `TRUE`, will also list dependencies of dependencies. When `check_updates = TRUE`, the recursive traversal follows only "strong" dependencies (Depends/Imports/LinkingTo), so Suggests are not expanded recursively.
`dev`	If `FALSE` (default), checks for updates in the R-multiverse or CRAN (stable releases). If `TRUE`, checks the Stan R-universe (dev versions). This is only cogent for Stan packages, and cannot compare two dev versions.
`check_updates`	Logical. If `FALSE`, skips checking for remote versions and only reports locally installed package versions.

Value

A data frame with columns:

package Package name.

remote Repository version (character, `NA` when not queried).

local Installed version (character, `"0"` if not installed).

behind Logical; `TRUE` when `remote` is newer than `local`.

Examples

```
## Not run:
# Full dependency check with remote versions
stanflow_deps(recursive = TRUE)

# Local-only inventory (fast, no network)
stanflow_deps(check_updates = FALSE)

## End(Not run)
```

`stanflow_logo` *Print the stanflow ASCII art logo*

Description

Adapted from `tidyverse::tidyverse_logo()`.

Usage

```
stanflow_logo()
```

Value

Invisibly returns the character vector that was printed.

Examples

```
stanflow_logo()
```

stanflow_update *Update stanflow packages*

Description

Checks for outdated Stan workflow packages and installs updates. This function requires an interactive R session for installation unless `dry_run = TRUE`. Dry runs check repositories and preview installs without prompting or installing packages. Adapted from `tidyverse::tidyverse_update()`.

Usage

```
stanflow_update(recursive = FALSE, dev = FALSE, dry_run = FALSE)
```

Arguments

`recursive`	If `TRUE`, will also list dependencies of dependencies. When `check_updates = TRUE`, the recursive traversal follows only "strong" dependencies (Depends/Imports/LinkingTo), so Suggests are not expanded recursively.
`dev`	If `FALSE` (default), checks for updates in the R-multiverse or CRAN (stable releases). If `TRUE`, checks the Stan R-universe (dev versions). This is only cogent for Stan packages, and cannot compare two dev versions.
`dry_run`	Logical. If `TRUE`, previews update steps without installing packages or prompting.

Value

Invisibly returns a data frame of outdated packages (same columns as `stanflow_deps`). Returns `NULL` invisibly when no updates are needed.

Examples

```
## Not run:
# Update direct dependencies only
stanflow_update()

# Update full dependency tree (including suggests)
stanflow_update(recursive = TRUE)

## End(Not run)
```

Index