# Package 'rstantools'

March 9, 2023

**Type** Package

**Title** Tools for Developing R Packages Interfacing with 'Stan'

**Version** 2.3.0

**Date** 2023-03-09

**Description** Provides various tools for developers of R packages interfacing with 'Stan' <https://mc-stan.org>, including functions to set up the required package structure, S3 generics and default methods to unify function naming across 'Stan'-based R packages, and vignettes with recommendations for developers.

**License** GPL (>= 3)

**URL** https://mc-stan.org/rstantools/, https://discourse.mc-stan.org/

**BugReports** https://github.com/stan-dev/rstantools/issues

**Encoding** UTF-8

**SystemRequirements** pandoc, C++14

**Imports** desc, stats, utils, Rcpp (>= 0.12.16), RcppParallel (>= 5.0.1)

**Suggests** rstan (>= 2.17.2), usethis (>= 1.5.1), testthat (>= 2.0.0), knitr, pkgbuild, pkgload, roxygen2 (>= 6.0.1), rmarkdown, rstudioapi

**RoxygenNote** 7.2.2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Jonah Gabry [aut, cre], Ben Goodrich [aut], Martin Lysy [aut], Andrew Johnson [aut], Hamada S. Badr [ctb], Marco Colombo [ctb], Stefan Siegert [ctb], Trustees of Columbia University [cph]

**Maintainer** Jonah Gabry <jsg2201@columbia.edu>

**Repository** CRAN

**Date/Publication** 2023-03-09 21:50:02 UTC

# R topics documented:



---

`rstantools-package` *Tools for Developing R Packages Interfacing with Stan*

---

## Description

*Stan Development Team*

The **rstantools** package provides various tools for developers of R packages interfacing with Stan (`https://mc-stan.org`), including functions to set up the required package structure, S3 generic methods to unify function naming across Stan-based R packages, and vignettes with guidelines for developers. To get started building a package see `rstan_create_package()`.

## See Also

- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

- After reading the guidelines for developers, if you have trouble setting up your package let us know on the the Stan Forums or at the **rstantools** GitHub issue tracker.

`bayes_R2` *Generic function and default method for Bayesian R-squared*

## Description

Generic function and default method for Bayesian version of R-squared for regression models. A generic for LOO-adjusted R-squared is also provided. See the bayes_R2.stanreg() method in the **rstanarm** package for an example of defining a method.

## Usage

```
bayes_R2(object, ...)

## Default S3 method:
bayes_R2(object, y, ...)

loo_R2(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |
| `y` | For the default method, a vector of `y` values the same length as the number of columns in the matrix used as `object`. |

## Value

`bayes_R2()` and `loo_R2()` methods should return a vector of length equal to the posterior sample size.

The default `bayes_R2()` method just takes `object` to be a matrix of y-hat values (one column per observation, one row per posterior draw) and `y` to be a vector with length equal to `ncol(object)`.

## References

Andrew Gelman, Ben Goodrich, Jonah Gabry, and Aki Vehtari (2018). R-squared for Bayesian regression models. *The American Statistician*, to appear. DOI: 10.1080/00031305.2018.1549100. (Preprint, Notebook)

## See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

## init_cpp *Register functions implemented in C++*

### Description

If you set up your package using `rstan_package_skeleton()` before version `1.2.1` of **rstantools** it may be necessary for you to call this function yourself in order to pass `R CMD check` in R `>= 3.4`. If you used `rstan_package_skeleton()` in **rstantools** version `1.2.1` or later then this has already been done automatically.

### Usage

```
init_cpp(name, path)
```

### Arguments

| | |
|---|---|
| `name` | The name of your package as a string. |
| `path` | The path to the root directory for your package as a string. If not specified it is assumed that this is already the current working directory. |

### Value

This function is only called for its side effect of writing the necessary `init.cpp` file to the package's `src/` directory.

## log_lik *Generic function for pointwise log-likelihood*

### Description

We define a new function `log_lik()` rather than a `stats::logLik()` method because (in addition to the conceptual difference) the documentation for `logLik()` states that the return value will be a single number, whereas `log_lik()` returns a matrix. See the log_lik.stanreg() method in the **rstanarm** package for an example of defining a method.

### Usage

```
log_lik(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |

### Value

`log_lik()` methods should return a $S$ by $N$ matrix, where $S$ is the size of the posterior sample (the number of draws from the posterior distribution) and $N$ is the number of data points.

### See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

### Examples

```
# See help("log_lik", package = "rstanarm")
```

---

## `loo-prediction` *Generic functions for LOO predictions*

---

### Description

See the methods in the **rstanarm** package for examples.

### Usage

```
loo_linpred(object, ...)

loo_predict(object, ...)

loo_predictive_interval(object, ...)

loo_pit(object, ...)

## Default S3 method:
loo_pit(object, y, lw, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |
| `y` | For the default method of `loo_pit()`, a vector of `y` values the same length as the number of columns in the matrix used as `object`. |
| `lw` | For the default method of `loo_pit()`, a matrix of log-weights of the same length as the number of columns in the matrix used as `object`. |

### Value

`loo_predict()`, `loo_linpred()`, and `loo_pit()` (probability integral transform) methods should return a vector with length equal to the number of observations in the data. `loo_predictive_interval()` methods should return a two-column matrix formatted in the same way as for `predictive_interval()`.

### See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

---

## nsamples — *Generic function for extracting the number of posterior samples*

---

### Description

Extract the number of posterior samples stored in a fitted Bayesian model.

### Usage

```
nsamples(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |

---

## posterior_epred — *Generic function for accessing the posterior distribution of the conditional expectation*

---

### Description

Extract the posterior draws of the conditional expectation. See the **rstanarm** package for an example.

### Usage

```
posterior_epred(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |

### Value

`posterior_epred()` methods should return a $D$ by $N$ matrix, where $D$ is the number of draws from the posterior distribution distribution and $N$ is the number of data points.

### See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

---

## `posterior_interval` *Generic function and default method for posterior uncertainty intervals*

---

### Description

These intervals are often referred to as credible intervals, but we use the term uncertainty intervals to highlight the fact that wider intervals correspond to greater uncertainty. See posterior_interval.stanreg() in the **rstanarm** package for an example.

### Usage

```
posterior_interval(object, ...)

## Default S3 method:
posterior_interval(object, prob = 0.9, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |
| `prob` | A number $p \in (0, 1)$ indicating the desired probability mass to include in the intervals. |

## Value

`posterior_interval()` methods should return a matrix with two columns and as many rows as model parameters (or a subset of parameters specified by the user). For a given value of `prob`, $p$, the columns correspond to the lower and upper $100p$\ have the names $100\alpha/2$\ $\alpha = 1 - p$. For example, if `prob=0.9` is specified (a 90\ `"95%"`, respectively.

The default method just takes `object` to be a matrix (one column per parameter) and computes quantiles, with `prob` defaulting to `0.9`.

## See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

## Examples

```
# Default method takes a numeric matrix (of posterior draws)
draws <- matrix(rnorm(100 * 5), 100, 5) # fake draws
colnames(draws) <- paste0("theta_", 1:5)
posterior_interval(draws)

# Also see help("posterior_interval", package = "rstanarm")
```

---

# posterior_linpred *Generic function for accessing the posterior distribution of the linear predictor*

---

## Description

Extract the posterior draws of the linear predictor, possibly transformed by the inverse-link function. See posterior_linpred.stanreg() in the **rstanarm** package for an example.

## Usage

```
posterior_linpred(object, transform = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `object` | The object to use. |
| `transform` | Should the linear predictor be transformed using the inverse-link function? The default is `FALSE`, in which case the untransformed linear predictor is returned. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |

### Value

`posterior_linpred()` methods should return a $D$ by $N$ matrix, where $D$ is the number of draws from the posterior distribution distribution and $N$ is the number of data points.

### See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

### Examples

```
# See help("posterior_linpred", package = "rstanarm")
```

---

## `posterior_predict` *Generic function for drawing from the posterior predictive distribution*

---

### Description

Draw from the posterior predictive distribution of the outcome. See posterior_predict.stanreg() in the **rstanarm** package for an example.

### Usage

```
posterior_predict(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |

### Value

`posterior_predict()` methods should return a $D$ by $N$ matrix, where $D$ is the number of draws from the posterior predictive distribution and $N$ is the number of data points being predicted per draw.

### See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

## Examples

```
# See help("posterior_predict", package = "rstanarm")
```

---

`predictive_error` *Generic function and default method for predictive errors*

---

## Description

Generic function and default method for computing predictive errors $y - y^{rep}$ (in-sample, for observed $y$) or $y - \tilde{y}$ (out-of-sample, for new or held-out $y$). See predictive_error.stanreg() in the **rstanarm** package for an example.

## Usage

```
predictive_error(object, ...)

## Default S3 method:
predictive_error(object, y, ...)
```

## Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |
| `y` | For the default method, a vector of `y` values the same length as the number of columns in the matrix used as `object`. |

## Value

`predictive_error()` methods should return a $D$ by $N$ matrix, where $D$ is the number of draws from the posterior predictive distribution and $N$ is the number of data points being predicted per draw.

The default method just takes `object` to be a matrix and `y` to be a vector.

## See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

### Examples

```
# default method
y <- rnorm(10)
ypred <- matrix(rnorm(500), 50, 10)
pred_errors <- predictive_error(ypred, y)
dim(pred_errors)
head(pred_errors)

# Also see help("predictive_error", package = "rstanarm")
```

---

## predictive_interval *Generic function for predictive intervals*

---

### Description

See predictive_interval.stanreg() in the **rstanarm** package for an example.

### Usage

```
predictive_interval(object, ...)

## Default S3 method:
predictive_interval(object, prob = 0.9, ...)
```

### Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |
| `prob` | A number $p \in (0, 1)$ indicating the desired probability mass to include in the intervals. |

### Value

`predictive_interval()` methods should return a matrix with two columns and as many rows as data points being predicted. For a given value of `prob`, $p$, the columns correspond to the lower and upper $100p\backslash 100(1 - \alpha/2)\backslash$ `prob=0.9` is specified (a $90\backslash$ would be `"5%"` and `"95%"`, respectively.

The default method just takes `object` to be a matrix and computes quantiles, with `prob` defaulting to `0.9`.

### See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

## Examples

```
# Default method takes a numeric matrix (of draws from posterior
# predictive distribution)
ytilde <- matrix(rnorm(100 * 5, sd = 2), 100, 5) # fake draws
predictive_interval(ytilde, prob = 0.8)

# Also see help("predictive_interval", package = "rstanarm")
```

---

`prior_summary` *Generic function for extracting information about prior distributions*

---

## Description

See prior_summary.stanreg() in the **rstanarm** package for an example.

## Usage

```
prior_summary(object, ...)

## Default S3 method:
prior_summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | The object to use. |
| `...` | Arguments passed to methods. See the methods in the **rstanarm** package for examples. |

## Value

`prior_summary()` methods should return an object containing information about the prior distribution(s) used for the given model. The structure of this object will depend on the method.

The default method just returns `object$prior.info`, which is `NULL` if there is no `'prior.info'` element.

## See Also

- The **rstanarm** package (mc-stan.org/rstanarm) for example methods (CRAN, GitHub).
- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.

## Examples

```
# See help("prior_summary", package = "rstanarm")
```

## rstan_config *Configure system files for compiling Stan source code*

### Description

Creates or update package-specific system files to compile `.stan` model files found in `inst/stan`.

### Usage

```
rstan_config(pkgdir = ".")
```

### Arguments

| | |
|---|---|
| `pkgdir` | Path to package root folder. |

### Details

The Stan source files for the package should be stored in:

- `inst/stan` for `.stan` files containing instructions to build a `stanmodel` object.
- `inst/stan/any_subfolder` for files to be included via the `#include "/my_subfolder/mylib.stan"` directive.
- `inst/stan/any_subfolder` for a `license.stan` file.
- `inst/include` for the `stan_meta_header.hpp` file, to be used for directly interacting with the Stan C++ libraries.

### Value

Invisibly, whether or not any files were added/removed/modified by the function.

## rstan_create_package *Create a new* R *package with compiled Stan programs*

### Description

The `rstan_create_package()` function helps get you started developing a new R package that interfaces with Stan via the **rstan** package. First the basic package structure is set up via `usethis::create_package()`. Then several adjustments are made so the package can include Stan programs that can be built into binary versions (i.e., pre-compiled Stan C++ code).

The **Details** section below describes the process and the **See Also** section provides links to recommendations for developers and a step-by-step walk-through.

As of version `2.0.0` of **rstantools** the `rstan_package_skeleton()` function is defunct and only `rstan_create_package()` is supported.

## Usage

```
rstan_create_package(
  path,
  fields = NULL,
  rstudio = TRUE,
  open = TRUE,
  stan_files = character(),
  roxygen = TRUE,
  travis = FALSE,
  license = TRUE,
  auto_config = TRUE
)
```

## Arguments

| | |
|---|---|
| path | The path to the new package to be created (terminating in the package name). |
| fields, rstudio, open | Same as usethis::create_package(). See the documentation for that function, especially the note in the **Description** section about the side effect of changing the active project. |
| stan_files | A character vector with paths to .stan files to include in the package. |
| roxygen | Should **roxygen2** be used for documentation? Defaults to TRUE. If so, a file 'R/pkgname-package.R" is added to the package with roxygen tags for the required import lines. See the **Note** section below for advice specific to the latest versions of **roxygen2**. |
| travis | Should a .travis.yml file be added to the package directory? This argument is now deprecated. We recommend using GitHub Actions to set up automated testings for your package. See https://github.com/r-lib/actions for useful templates. |
| license | Logical or character; whether or not to paste the contents of a license.stan file at the top of all Stan code, or path to such a file. If TRUE (the default) adds the GPL (>= 3) license (see **Details**). |
| auto_config | Whether to automatically configure Stan functionality whenever the package gets installed (see **Details**). Defaults to TRUE. |

## Details

This function first creates a regular R package using usethis::create_package(), then adds the infrastructure required to compile and export stanmodel objects. In the package root directory, the user's Stan source code is located in:

```
inst/
  |_stan/
  |   |_include/
  |
  |_include/
```

All .stan files containing instructions to build a stanmodel object must be placed in inst/stan. Other .stan files go in any stan/ subdirectory, to be invoked by Stan's #include mechanism, e.g.,

```
#include "include/mylib.stan"
#include "data/preprocess.stan"
```

See **rstanarm** for many examples.

The folder `inst/include` is for all user C++ files associated with the Stan programs. In this folder, the only file to directly interact with the Stan C++ library is `stan_meta_header.hpp`; all other `#include` directives must be channeled through here.

The final step of the package creation is to invoke `rstan_config()`, which creates the following files for interfacing with Stan objects from R:

- `src` contains the `stan_ModelName{.cc/.hpp}` pairs associated with all `ModelName.stan` files in `inst/stan` which define `stanmodel` objects.
- `src/Makevars[.win]` which link to the `StanHeaders` and Boost (`BH`) libraries.
- `R/stanmodels.R` loads the C++ modules containing the `stanmodel` class definitions, and assigns an R instance of each `stanmodel` object to a `stanmodels` list (with names corresponding to the names of the Stan files).

When `auto_config = TRUE`, a `configure[.win]` file is added to the package, calling `rstan_config()` whenever the package is installed. Consequently, the package must list **rstantools** in the `DESCRIPTION` Imports field for this mechanism to work. Setting `auto_config = FALSE` removes the package's dependency on **rstantools**, but the package then must be manually configured by running `rstan_config()` whenever `stanmodel` files in `inst/stan` are added, removed, or modified.

In order to enable Stan functionality, **rstantools** copies some files to your package. Since these files are licensed as GPL= 3, the same license applies to your package should you choose todistribute it. Even if you don't use **rstantools** to createyour package, it is likely that you will be linking to **Rcpp** toexport the Stan C++ stanmodel objects to R. Since**Rcpp** is released under GPL >= 2, the same license would applyto your package upon distribution.

Authors willing to license their Stan programs of general interest under the GPL are invited to contribute their `.stan` files and supporting R code to the **rstanarm** package.

### Using the pre-compiled Stan programs in your package

The `stanmodel` objects corresponding to the Stan programs included with your package are stored in a list called `stanmodels`. To run one of the Stan programs from within an R function in your package just pass the appropriate element of the `stanmodels` list to one of the **rstan** functions for model fitting (e.g., `sampling()`). For example, for a Stan program `"foo.stan"` you would use `rstan::sampling(stanmodels$foo, ...)`.

### Note

For **devtools** users, because of changes in the latest versions of **roxygen2** it may be necessary to run `pkgbuild::compile_dll()` once before `devtools::document()` will work.

### See Also

- `use_rstan()` for adding Stan functionality to an existing R package and `rstan_config()` for updating an existing package when its Stan files are changed.
- The **rstanarm** package repository on GitHub.

- Guidelines and recommendations for developers of R packages interfacing with Stan and a demonstration getting a simple package working can be found in the vignettes included with **rstantools** and at mc-stan.org/rstantools/articles.
- After reading the guidelines for developers, if you have trouble setting up your package let us know on the the Stan Forums or at the **rstantools** GitHub issue tracker.

---

`use_rstan` *Add Stan infrastructure to an existing package*

---

## Description

Add Stan infrastructure to an existing R package. To create a *new* package containing Stan programs use `rstan_create_package()` instead.

## Usage

```
use_rstan(pkgdir = ".", license = TRUE, auto_config = TRUE)
```

## Arguments

| | |
|---|---|
| `pkgdir` | Path to package root folder. |
| `license` | Logical or character; whether or not to paste the contents of a `license.stan` file at the top of all Stan code, or path to such a file. If `TRUE` (the default) adds the `GPL (>= 3)` license (see **Details**). |
| `auto_config` | Whether to automatically configure Stan functionality whenever the package gets installed (see **Details**). Defaults to `TRUE`. |

## Details

Prepares a package to compile and use Stan code by performing the following steps:

1. Create `inst/stan` folder where all `.stan` files defining Stan models should be stored.
2. Create `inst/stan/include` where optional `license.stan` file is stored.
3. Create `inst/include/stan_meta_header.hpp` to include optional header files used by Stan code.
4. Create `src` folder (if it doesn't exist) to contain the Stan C++ code.
5. Create `R` folder (if it doesn't exist) to contain wrapper code to expose Stan C++ classes to R.
6. Update `DESCRIPTION` file to contain all needed dependencies to compile Stan C++ code.
7. If `NAMESPACE` file is generic (i.e., created by `rstan_create_package()`), append `import(Rcpp, methods)`, `importFrom(rstan, sampling)`, and `useDynLib` directives. If `NAMESPACE` is not generic, display message telling user what to add to `NAMESPACE` for themselves.

When `auto_config = TRUE`, a `configure[.win]` file is added to the package, calling `rstan_config()` whenever the package is installed. Consequently, the package must list **rstantools** in the `DESCRIPTION` Imports field for this mechanism to work. Setting `auto_config = FALSE` removes the package's dependency on **rstantools**, but the package then must be manually configured by running `rstan_config()` whenever `stanmodel` files in `inst/stan` are added, removed, or modified.

## Value

Invisibly, `TRUE` or `FALSE` indicating whether or not any files or folders where created or modified.

## Using the pre-compiled Stan programs in your package

The `stanmodel` objects corresponding to the Stan programs included with your package are stored in a list called `stanmodels`. To run one of the Stan programs from within an R function in your package just pass the appropriate element of the `stanmodels` list to one of the **rstan** functions for model fitting (e.g., `sampling()`). For example, for a Stan program `"foo.stan"` you would use `rstan::sampling(stanmodels$foo, ...)`.

# Index