

# Package ‘jagstargets’

May 3, 2022

**Title** Targets for JAGS Workflows

**Description** Bayesian data analysis usually incurs long runtimes and cumbersome custom code.

A pipeline toolkit tailored to Bayesian statisticians,

the 'jagstargets' R package leverages

'targets' and 'R2jags' to ease this burden.

'jagstargets' makes it super easy to set up scalable

JAGS pipelines that automatically parallelize the computation

and skip expensive steps when the results are already up to date.

Minimal custom code is required, and there is no need to manually

configure branching, so usage is much easier than 'targets' alone.

For the underlying methodology, please refer

to the documentation of 'targets' <[doi:10.21105/joss.02959](https://doi.org/10.21105/joss.02959)> and 'JAGS'

(Plummer 2003) <[https:](https://www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf)

[//www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf](https://www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf)>.

**Version** 1.0.2

**License** MIT + file LICENSE

**URL** <https://docs.ropensci.org/jagstargets/>,

<https://github.com/ropensci/jagstargets>

**BugReports** <https://github.com/ropensci/jagstargets/issues>

**Depends** R (>= 3.5.0)

**Imports** coda (>= 0.19.4), digest (>= 0.6.25), fst (>= 0.9.2),

posterior (>= 1.0.1), purrr (>= 0.3.4), qs (>= 0.23.2), R2jags

(>= 0.6.1), rjags (>= 4.10), rlang (>= 0.4.10), stats, targets

(>= 0.12.0), tarchetypes (>= 0.6.0), tibble (>= 3.0.1), tools,

utils, withr (>= 2.1.2),

**Suggests** dplyr (>= 1.0.2), fs (>= 1.5.0), knitr (>= 1.30), R.utils (>=

2.10.1), rmarkdown (>= 2.3), testthat (>= 3.0.0), tidyr (>=

1.1.2), visNetwork (>= 2.0.9)

**SystemRequirements** JAGS 4.x.y (<https://mcmc-jags.sourceforge.net>)

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.1.2**VignetteBuilder** knitr**Config/testthat/edition** 3**NeedsCompilation** no

**Author** William Michael Landau [aut, cre]  
 (<<https://orcid.org/0000-0003-1878-3253>>),  
 David Lawrence Miller [rev],  
 Eli Lilly and Company [cph]

**Maintainer** William Michael Landau <will.landau@gmail.com>**Repository** CRAN**Date/Publication** 2022-05-03 15:00:02 UTC

## R topics documented:

|                                 |           |
|---------------------------------|-----------|
| jagstargets-package . . . . .   | 2         |
| tar_jags . . . . .              | 3         |
| tar_jags_example_data . . . . . | 8         |
| tar_jags_example_file . . . . . | 10        |
| tar_jags_rep_dic . . . . .      | 10        |
| tar_jags_rep_draws . . . . .    | 16        |
| tar_jags_rep_summary . . . . .  | 21        |
| <b>Index</b>                    | <b>28</b> |

---

jagstargets-package    *jagstargets: Targets for JAGS Workflows*

---

## Description

Bayesian data analysis usually incurs long runtimes and cumbersome custom code. A pipeline toolkit tailored to Bayesian statisticians, the `jagstargets` R package leverages `targets` and `R2jags` to ease this burden. `jagstargets` makes it super easy to set up scalable JAGS pipelines that automatically parallelize the computation and skip expensive steps when the results are already up to date. Minimal custom code is required, and there is no need to manually configure branching, so usage is much easier than `targets` alone.

## See Also

<https://docs.ropensci.org/jagstargets/>, `tar_jags()`

---

tar\_jags

*One MCMC per model with multiple outputs*

---

## Description

Targets to run a JAGS model once with MCMC and save multiple outputs.

## Usage

```
tar_jags(  
  name,  
  jags_files,  
  parameters.to.save,  
  data = list(),  
  summaries = list(),  
  summary_args = list(),  
  n.cluster = 1,  
  n.chains = 3,  
  n.iter = 2000,  
  n.burnin = as.integer(n.iter/2),  
  n.thin = 1,  
  jags.module = c("glm", "dic"),  
  inits = NULL,  
  RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry", "Super-Duper",  
    "Mersenne-Twister"),  
  jags.seed = 1,  
  stdout = NULL,  
  stderr = NULL,  
  progress.bar = "text",  
  refresh = 0,  
  draws = TRUE,  
  summary = TRUE,  
  dic = TRUE,  
  tidy_eval = targets::tar_option_get("tidy_eval"),  
  packages = targets::tar_option_get("packages"),  
  library = targets::tar_option_get("library"),  
  format = "qs",  
  format_df = "fst_tbl",  
  repository = targets::tar_option_get("repository"),  
  error = targets::tar_option_get("error"),  
  memory = targets::tar_option_get("memory"),  
  garbage_collection = targets::tar_option_get("garbage_collection"),  
  deployment = targets::tar_option_get("deployment"),  
  priority = targets::tar_option_get("priority"),  
  resources = targets::tar_option_get("resources"),  
  storage = targets::tar_option_get("storage"),  
  retrieval = targets::tar_option_get("retrieval"),
```

```

  cue = targets::tar_option_get("cue")
)

```

### Arguments

|                    |   |
|--------------------|---|
| name               | Symbol, base name for the collection of targets. Serves as a prefix for target names.   |
| jags_files         | Character vector of JAGS model files. If you supply multiple files, each model will run on the one shared dataset generated by the code in data. If you supply an unnamed vector, <code>tools::file_path_sans_ext(basename(jags_files))</code> will be used as target name suffixes. If <code>jags_files</code> is a named vector, the suffixed will come from <code>names(jags_files)</code> . |
| parameters.to.save | Model parameters to save, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| data               | Code to generate the data list for the JAGS model. Optionally include a <code>.join_data</code> element to join parts of the data to correspondingly named parameters in the summary output. See the vignettes for details.   |
| summaries          | List of summary functions passed to <code>...</code> in <code>posterior::summarize_draws()</code> through <code>\$summary()</code> on the <code>CmdStanFit</code> object.   |
| summary_args       | List of summary function arguments passed to <code>.args</code> in <code>posterior::summarize_draws()</code> through <code>\$summary()</code> on the <code>CmdStanFit</code> object.  |
| n.cluster          | Number of parallel processes, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| n.chains           | Number of MCMC chains, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.iter             | Number if iterations (including warmup), passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.burnin           | Number of warmup iterations, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.thin             | Thinning interval, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| jags.module        | Character vector of JAGS modules to load, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| inits              | Initial values of model parameters, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |

|              |   |
|--------------|---|
| RNGname      | Choice of random number generator, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| jags.seed    | Seeds to apply to JAGS, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| stdout       | Character of length 1, file path to write the stdout stream of the model when it runs. Set to <code>NULL</code> to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stdout. Does not apply to messages, warnings, or errors.   |
| stderr       | Character of length 1, file path to write the stderr stream of the model when it runs. Set to <code>NULL</code> to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stderr. Does not apply to messages, warnings, or errors.   |
| progress.bar | Type of progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| refresh      | Frequency for refreshing the progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| draws        | Logical, whether to create a target for posterior draws. Saves draws as a compressed <code>posterior::as_draws_df()</code> tibble. Convenient, but duplicates storage.  |
| summary      | Logical, whether to create a target to store a small data frame of posterior summary statistics and convergence diagnostics.  |
| dic          | Logical, whether to create a target with deviance information criterion (DIC) results.  |
| tidy_eval    | Logical, whether to enable tidy evaluation when interpreting command and pattern. If <code>TRUE</code> , you can use the "bang-bang" operator <code>!!</code> to programmatically insert the values of global objects.  |
| packages     | Character vector of packages to load right before the target builds or the output data is reloaded for downstream targets. Use <code>tar_option_set()</code> to set packages globally for all subsequent targets you define.  |
| library      | Character vector of library paths to try when loading packages.   |
| format       | Character of length 1, storage format of the non-data-frame targets such as the JAGS data and any JAGS fit objects. Please choose an all-purpose format such as <code>"qs"</code> or <code>"aws_qs"</code> rather than a file format like <code>"file"</code> or a data frame format like <code>"parquet"</code> . For more on storage formats, see the help file of <code>targets::tar_target()</code> . |
| format_df    | Character of length 1, storage format of the data frame targets such as posterior draws. We recommend efficient data frame formats such as <code>"feather"</code> or <code>"aws_parquet"</code> . For more on storage formats, see the help file of <code>targets::tar_target()</code> .  |
| repository   | Character of length 1, remote repository for target storage. Choices: <ul style="list-style-type: none"><li>• <code>"local"</code>: file system of the local machine.</li></ul>   |

|                    |  |
|--------------------|--|
|                    | <ul style="list-style-type: none"> <li>• "aws": Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the endpoint argument of <code>tar_resources_aws()</code>, but versioning capabilities may be lost in doing so. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> <li>• "gcp": Google Cloud Platform storage bucket. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> </ul>   |
| error              | <p>Character of length 1, what to do if the target stops and throws an error. Options:</p> <ul style="list-style-type: none"> <li>• "stop": the whole pipeline stops and throws an error.</li> <li>• "continue": the whole pipeline keeps going.</li> <li>• "abridge": any currently running targets keep running, but no new targets launch after that. (Visit <a href="https://books.ropensci.org/targets/debugging.html">https://books.ropensci.org/targets/debugging.html</a> to learn how to debug targets using saved workspaces.)</li> <li>• "null": The errored target continues and returns NULL. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline.</li> </ul>  |
| memory             | <p>Character of length 1, memory strategy. If "persistent", the target stays in memory until the end of the pipeline (unless storage is "worker", in which case targets unloads the value from memory right after storing it in order to avoid sending copious data over a network). If "transient", the target gets unloaded after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value. For cloud-based dynamic files such as <code>format = "aws_file"</code>, this memory strategy applies to temporary local copies of the file in <code>_targets/scratch/</code>: "persistent" means they remain until the end of the pipeline, and "transient" means they get deleted from the file system as soon as possible. The former conserves bandwidth, and the latter conserves local storage.</p> |
| garbage_collection | <p>Logical, whether to run <code>base::gc()</code> just before the target runs.</p>  |
| deployment         | <p>Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code>. If "worker", the target builds on a parallel worker. If "main", the target builds on the host machine / process managing the pipeline.</p>  |
| priority           | <p>Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get built earlier (and polled earlier in <code>tar_make_future()</code>).</p>  |
| resources          | <p>Object returned by <code>tar_resources()</code> with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See <code>tar_resources()</code> for details.</p>  |
| storage            | <p>Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code>. Must be one of the following values:</p> <ul style="list-style-type: none"> <li>• "main": the target's return value is sent back to the host machine and saved/uploaded locally.</li> <li>• "worker": the worker saves/uploads the value.</li> </ul>  |

- "none": almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language. If you do use it, then the return value of the target is totally ignored when the target ends, but each downstream target still attempts to load the data file (except when `retrieval = "none"`).

If you select `storage = "none"`, then the return value of the target's command is ignored, and the data is not saved automatically. As with dynamic files (`format = "file"` or `"aws_file"`) it is the responsibility of the user to write to `tar_path()` from inside the target. An example target could look something like `tar_target(x, saveRDS("value", tar_path(create_dir = TRUE)); "ignored", storage = "none")`.

The distinguishing feature of `storage = "none"` (as opposed to `format = "file"` or `"aws_file"`) is that in the general case, downstream targets will automatically try to load the data from the data store as a dependency. As a corollary, `storage = "none"` is completely unnecessary if `format = "file"` or `"aws_file"`.

|           |   |
|-----------|---|
| retrieval | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . Must be one of the following values: <ul style="list-style-type: none"> <li>• "main": the target's dependencies are loaded on the host machine and sent to the worker before the target builds.</li> <li>• "worker": the worker loads the targets dependencies.</li> <li>• "none": the dependencies are not loaded at all. This choice is almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language.</li> </ul> |
| cue       | An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.   |

## Details

The MCMC targets use `R2jags::jags()` if `n.cluster` is 1 and `R2jags::jags.parallel()` otherwise. Most arguments to `tar_jags()` are forwarded to these functions.

## Value

`tar_jags()` returns list of target objects. See the "Target objects" section for background. The target names use the name argument as a prefix, and the individual elements of `jags_files` appear in the suffixes where applicable. As an example, the specific target objects returned by `tar_jags(name = x, jags_files = "y.jags", ...)` returns a list of targets: `tar_target()` objects:

- `x_file_y`: reproducibly track the JAGS model file. Returns a character vector of length 1 with the path to the JAGS model file.
- `x_lines_y`: read the contents of the JAGS model file for safe transport to parallel workers. Returns a character vector of lines in the model file.
- `x_data`: run the R expression in the data argument to produce a JAGS dataset for the model. Returns a JAGS data list.
- `x_mcmc_y`: run MCMC on the model and dataset. Returns an `rjags` object from `R2jags` with all the MCMC results.

- `x_draws_y`: extract posterior samples from `x_mcmc_y`. Returns a tidy data frame of MCMC draws. Omitted if `draws = FALSE`.
- `x_summary_y`: extract posterior summaries from `x_mcmc_y`. Returns a tidy data frame of MCMC draws. Omitted if `summary = FALSE`.
- `x_dic`: extract deviance information criterion (DIC) info from `x_mcmc_y`. Returns a tidy data frame of DIC info. Omitted if `dic = FALSE`.

## Target objects

Most `stantargets` functions are target factories, which means they return target objects or lists of target objects. Target objects represent skippable steps of the analysis pipeline as described at <https://books.ropensci.org/targets/>. Please read the walkthrough at <https://books.ropensci.org/targets/walkthrough.html> to understand the role of target objects in analysis pipelines.

For developers, <https://wlandau.github.io/targetopia/contributing.html#target-factories> explains target factories (functions like this one which generate targets) and the design specification at <https://books.ropensci.org/targets-design/> details the structure and composition of target objects.

## Examples

```
if (requireNamespace("R2jags", quietly = TRUE)) {
  targets::tar_dir({ # tar_dir() runs code from a temporary directory.
    targets::tar_script({
      library(jagstargets)
      # Do not use a temp file for a real project
      # or else your targets will always rerun.
      tmp <- tempfile(pattern = "", fileext = ".jags")
      tar_jags_example_file(tmp)
      list(
        tar_jags(
          your_model,
          jags_files = tmp,
          data = tar_jags_example_data(),
          parameters.to.save = "beta",
          stdout = R.utils::nullfile(),
          stderr = R.utils::nullfile()
        )
      )
    }, ask = FALSE)
  targets::tar_make()
})
}
```

---

`tar_jags_example_data` *Simulate example JAGS data.*

---



## Description

An example dataset compatible with the model file from `tar_jags_example_file()`. The output has a `.join_data` element so the true value of beta from the simulation is automatically appended to the beta rows of the summary output.

## Usage

```
tar_jags_example_data(n = 10L)
```

## Arguments

`n` Integer of length 1, number of data points.

## Format

A list with the following elements:

- `n`: integer, number of data points.
- `x`: numeric, covariate vector.
- `y`: numeric, response variable.
- `true_beta`: numeric of length 1, value of the regression coefficient beta used in simulation.
- `.join_data`: a list of simulated values to be appended to as a `.join_data` column in the output of targets generated by functions such as `tar_jags_rep_summary()`. Contains the regression coefficient beta (numeric of length 1) and prior predictive data y (numeric vector).

## Details

The `tar_jags_example_data()` function draws a JAGS dataset from the prior predictive distribution of the model from `tar_jags_example_file()`. First, the regression coefficient beta is drawn from its standard normal prior, and the covariate x is computed. Then, conditional on the beta draws and the covariate, the response vector y is drawn from its  $\text{Normal}(x * \text{beta}, 1)$  likelihood.

## Value

List, dataset compatible with the model file from `tar_jags_example_file()`. The output has a `.join_data` element so the true value of beta from the simulation is automatically appended to the beta rows of the summary output.

## Examples

```
tar_jags_example_data()
```

---

tar\_jags\_example\_file *Write an example JAGS model file.*

---

**Description**

Overwrites the file at path with a built-in example JAGS model file.

**Usage**

```
tar_jags_example_file(path = tempfile(pattern = "", fileext = ".jags"))
```

**Arguments**

path                    Character of length 1, file path to write the model file.

**Value**

NULL (invisibly).

**Examples**

```
path <- tempfile(pattern = "", fileext = ".jags")
tar_jags_example_file(path = path)
writeLines(readLines(path))
```

---

tar\_jags\_rep\_dic            *Tidy DIC output from multiple MCMCs per model*

---

**Description**

Run multiple MCMCs on simulated datasets and return DIC and the effective number of parameters for each run.

**Usage**

```
tar_jags_rep_dic(
  name,
  jags_files,
  parameters.to.save,
  data = list(),
  batches = 1L,
  reps = 1L,
  combine = TRUE,
  n.cluster = 1,
  n.chains = 3,
  n.iter = 2000,
```

```

n.burnin = as.integer(n.iter/2),
n.thin = 1,
jags.module = c("glm", "dic"),
inits = NULL,
RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry", "Super-Duper",
  "Mersenne-Twister"),
jags.seed = 123,
stdout = NULL,
stderr = NULL,
progress.bar = "text",
refresh = 0,
tidy_eval = targets::tar_option_get("tidy_eval"),
packages = targets::tar_option_get("packages"),
library = targets::tar_option_get("library"),
format = "qs",
format_df = "fst_tbl",
repository = targets::tar_option_get("repository"),
error = targets::tar_option_get("error"),
memory = targets::tar_option_get("memory"),
garbage_collection = targets::tar_option_get("garbage_collection"),
deployment = targets::tar_option_get("deployment"),
priority = targets::tar_option_get("priority"),
resources = targets::tar_option_get("resources"),
storage = targets::tar_option_get("storage"),
retrieval = targets::tar_option_get("retrieval"),
cue = targets::tar_option_get("cue")
)

```

## Arguments

|                    |   |
|--------------------|---|
| name               | Symbol, base name for the collection of targets. Serves as a prefix for target names.   |
| jags_files         | Character vector of JAGS model files. If you supply multiple files, each model will run on the one shared dataset generated by the code in data. If you supply an unnamed vector, <code>tools::file_path_sans_ext(basename(jags_files))</code> will be used as target name suffixes. If <code>jags_files</code> is a named vector, the suffixed will come from <code>names(jags_files)</code> . |
| parameters.to.save | Model parameters to save, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| data               | Code to generate the data list for the JAGS model. Optionally include a <code>.join_data</code> element to join parts of the data to correspondingly named parameters in the summary output. See the vignettes for details.   |
| batches            | Number of batches. Each batch runs a model <code>reps</code> times.   |
| reps               | Number of replications per batch. Ideally, each rep should produce its own random dataset using the code supplied to <code>data</code> .  |

|              |   |
|--------------|---|
| combine      | Logical, whether to create a target to combine all the model results into a single data frame downstream. Convenient, but duplicates data.  |
| n.cluster    | Number of parallel processes, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.              |
| n.chains     | Number of MCMC chains, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                     |
| n.iter       | Number of iterations (including warmup), passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.burnin     | Number of warmup iterations, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.               |
| n.thin       | Thinning interval, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                         |
| jags.module  | Character vector of JAGS modules to load, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| inits        | Initial values of model parameters, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.        |
| RNGname      | Choice of random number generator, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.         |
| jags.seed    | Seeds to apply to JAGS, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                    |
| stdout       | Character of length 1, file path to write the stdout stream of the model when it runs. Set to <code>NULL</code> to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stdout. Does not apply to messages, warnings, or errors.         |
| stderr       | Character of length 1, file path to write the stderr stream of the model when it runs. Set to <code>NULL</code> to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stderr. Does not apply to messages, warnings, or errors.         |
| progress.bar | Type of progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                      |
| refresh      | Frequency for refreshing the progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details. |
| tidy_eval    | Logical, whether to enable tidy evaluation when interpreting command and pattern. If <code>TRUE</code> , you can use the "bang-bang" operator <code>!!</code> to programmatically insert the values of global objects.  |

|                    |   |
|--------------------|---|
| packages           | Character vector of packages to load right before the target builds or the output data is reloaded for downstream targets. Use <code>tar_option_set()</code> to set packages globally for all subsequent targets you define.  |
| library            | Character vector of library paths to try when loading packages.   |
| format             | Character of length 1, storage format of the data frames of posterior summaries and other data frames returned by targets. We recommend efficient data frame formats such as "feather" or "aws_parquet". For more on storage formats, see the help file of <code>targets::tar_target()</code> .   |
| format_df          | Character of length 1, storage format of the data frame targets such as posterior draws. We recommend efficient data frame formats such as "feather" or "aws_parquet". For more on storage formats, see the help file of <code>targets::tar_target()</code> .   |
| repository         | Character of length 1, remote repository for target storage. Choices: <ul style="list-style-type: none"> <li>• "local": file system of the local machine.</li> <li>• "aws": Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the <code>endpoint</code> argument of <code>tar_resources_aws()</code>, but versioning capabilities may be lost in doing so. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> <li>• "gcp": Google Cloud Platform storage bucket. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> </ul>   |
| error              | Character of length 1, what to do if the target stops and throws an error. Options: <ul style="list-style-type: none"> <li>• "stop": the whole pipeline stops and throws an error.</li> <li>• "continue": the whole pipeline keeps going.</li> <li>• "abridge": any currently running targets keep running, but no new targets launch after that. (Visit <a href="https://books.ropensci.org/targets/debugging.html">https://books.ropensci.org/targets/debugging.html</a> to learn how to debug targets using saved workspaces.)</li> <li>• "null": The errored target continues and returns NULL. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline.</li> </ul>  |
| memory             | Character of length 1, memory strategy. If "persistent", the target stays in memory until the end of the pipeline (unless storage is "worker", in which case targets unloads the value from memory right after storing it in order to avoid sending copious data over a network). If "transient", the target gets unloaded after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value. For cloud-based dynamic files such as <code>format = "aws_file"</code> , this memory strategy applies to temporary local copies of the file in <code>_targets/scratch/</code> : "persistent" means they remain until the end of the pipeline, and "transient" means they get deleted from the file system as soon as possible. The former conserves bandwidth, and the latter conserves local storage. |
| garbage_collection | Logical, whether to run <code>base::gc()</code> just before the target runs.  |
| deployment         | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . If "worker", the target builds on a parallel worker. If "main", the target builds on the host machine / process managing the pipeline.   |

|           |  |
|-----------|--|
| priority  | Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get built earlier (and polled earlier in <code>tar_make_future()</code> ).  |
| resources | Object returned by <code>tar_resources()</code> with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See <code>tar_resources()</code> for details.   |
| storage   | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . Must be one of the following values: <ul style="list-style-type: none"> <li>• "main": the target's return value is sent back to the host machine and saved/uploaded locally.</li> <li>• "worker": the worker saves/uploads the value.</li> <li>• "none": almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language. If you do use it, then the return value of the target is totally ignored when the target ends, but each downstream target still attempts to load the data file (except when <code>retrieval = "none"</code>).</li> </ul> <p>If you select <code>storage = "none"</code>, then the return value of the target's command is ignored, and the data is not saved automatically. As with dynamic files (<code>format = "file"</code> or <code>"aws_file"</code>) it is the responsibility of the user to write to <code>tar_path()</code> from inside the target. An example target could look something like <code>tar_target(x, saveRDS("value", tar_path(create_dir = TRUE)); "ignored", storage = "none")</code>.</p> <p>The distinguishing feature of <code>storage = "none"</code> (as opposed to <code>format = "file"</code> or <code>"aws_file"</code>) is that in the general case, downstream targets will automatically try to load the data from the data store as a dependency. As a corollary, <code>storage = "none"</code> is completely unnecessary if <code>format</code> is <code>"file"</code> or <code>"aws_file"</code>.</p> |
| retrieval | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . Must be one of the following values: <ul style="list-style-type: none"> <li>• "main": the target's dependencies are loaded on the host machine and sent to the worker before the target builds.</li> <li>• "worker": the worker loads the targets dependencies.</li> <li>• "none": the dependencies are not loaded at all. This choice is almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language.</li> </ul>  |
| cue       | An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.  |

### Details

The MCMC targets use `R2jags::jags()` if `n.cluster` is 1 and `R2jags::jags.parallel()` otherwise. Most arguments to `tar_jags()` are forwarded to these functions.

### Value

`tar_jags_rep_dic()` returns list of target objects. See the "Target objects" section for background. The target names use the name argument as a prefix, and the individual elements of `jags_files`

appear in the suffixes where applicable. As an example, the specific target objects returned by `tar_jags_rep_dic(name = x, jags_files = "y.jags")` are as follows.

- `x_file_y`: reproducibly track the JAGS model file. Returns a character vector of length 1 with the path to the JAGS model file.
- `x_lines_y`: read the contents of the JAGS model file for safe transport to parallel workers. Returns a character vector of lines in the model file.
- `x_data`: use dynamic branching to generate multiple JAGS datasets from the R expression in the data argument. Each dynamic branch returns a batch of JAGS data lists.
- `x_y`: run JAGS on each dataset from `x_data`. Each dynamic branch returns a tidy data frame of DIC results for each batch of data.
- `x`: combine all the batches from `x_y` into a non-dynamic target. Suppressed if `combine` is `FALSE`. Returns a long tidy data frame with all DIC info from all the branches of `x_y`.

## Target objects

Most `stantargets` functions are target factories, which means they return target objects or lists of target objects. Target objects represent skippable steps of the analysis pipeline as described at <https://books.ropensci.org/targets/>. Please read the walkthrough at <https://books.ropensci.org/targets/walkthrough.html> to understand the role of target objects in analysis pipelines.

For developers, <https://wlandau.github.io/targetopia/contributing.html#target-factories> explains target factories (functions like this one which generate targets) and the design specification at <https://books.ropensci.org/targets-design/> details the structure and composition of target objects.

## Examples

```
if (requireNamespace("R2jags", quietly = TRUE)) {
  targets::tar_dir({ # tar_dir() runs code from a temporary directory.
    targets::tar_script({
      library(jagstargets)
      # Do not use a temp file for a real project
      # or else your targets will always rerun.
      tmp <- tempfile(pattern = "", fileext = ".jags")
      tar_jags_example_file(tmp)
      list(
        tar_jags_rep_dic(
          your_model,
          jags_files = tmp,
          data = tar_jags_example_data(),
          parameters.to.save = "beta",
          batches = 2,
          reps = 2,
          stdout = R.utils::nullfile(),
          stderr = R.utils::nullfile()
        )
      )
    }, ask = FALSE)
  targets::tar_make()
```

```

  })
}

```

---

tar\_jags\_rep\_draws      *Tidy posterior draws from multiple MCMCs per model*

---

## Description

Run multiple MCMCs on simulated datasets and return posterior samples and the effective number of parameters for each run.

## Usage

```

tar_jags_rep_draws(
  name,
  jags_files,
  parameters.to.save,
  data = list(),
  batches = 1L,
  reps = 1L,
  combine = FALSE,
  n.cluster = 1,
  n.chains = 3,
  n.iter = 2000,
  n.burnin = as.integer(n.iter/2),
  n.thin = 1,
  jags.module = c("glm", "dic"),
  inits = NULL,
  RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry", "Super-Duper",
    "Mersenne-Twister"),
  jags.seed = 123,
  stdout = NULL,
  stderr = NULL,
  progress.bar = "text",
  refresh = 0,
  tidy_eval = targets::tar_option_get("tidy_eval"),
  packages = targets::tar_option_get("packages"),
  library = targets::tar_option_get("library"),
  format = "qs",
  format_df = "fst_tbl",
  repository = targets::tar_option_get("repository"),
  error = targets::tar_option_get("error"),
  memory = "transient",
  garbage_collection = targets::tar_option_get("garbage_collection"),
  deployment = targets::tar_option_get("deployment"),
  priority = targets::tar_option_get("priority"),
  resources = targets::tar_option_get("resources"),

```



```

storage = targets::tar_option_get("storage"),
retrieval = targets::tar_option_get("retrieval"),
cue = targets::tar_option_get("cue")
)

```

## Arguments

|                    |   |
|--------------------|---|
| name               | Symbol, base name for the collection of targets. Serves as a prefix for target names.   |
| jags_files         | Character vector of JAGS model files. If you supply multiple files, each model will run on the one shared dataset generated by the code in data. If you supply an unnamed vector, <code>tools::file_path_sans_ext(basename(jags_files))</code> will be used as target name suffixes. If <code>jags_files</code> is a named vector, the suffixed will come from <code>names(jags_files)</code> . |
| parameters.to.save | Model parameters to save, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| data               | Code to generate the data list for the JAGS model. Optionally include a <code>.join_data</code> element to join parts of the data to correspondingly named parameters in the summary output. See the vignettes for details.   |
| batches            | Number of batches. Each batch runs a model <code>reps</code> times.   |
| reps               | Number of replications per batch. Ideally, each rep should produce its own random dataset using the code supplied to <code>data</code> .  |
| combine            | Logical, whether to create a target to combine all the model results into a single data frame downstream. Convenient, but duplicates data.  |
| n.cluster          | Number of parallel processes, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| n.chains           | Number of MCMC chains, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.iter             | Number if iterations (including warmup), passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.burnin           | Number of warmup iterations, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| n.thin             | Thinning interval, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| jags.module        | Character vector of JAGS modules to load, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |

|              |   |
|--------------|---|
| inits        | Initial values of model parameters, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| RNGname      | Choice of random number generator, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| jags.seed    | Seeds to apply to JAGS, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| stdout       | Character of length 1, file path to write the stdout stream of the model when it runs. Set to <code>NULL</code> to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stdout. Does not apply to messages, warnings, or errors.   |
| stderr       | Character of length 1, file path to write the stderr stream of the model when it runs. Set to <code>NULL</code> to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stderr. Does not apply to messages, warnings, or errors.   |
| progress.bar | Type of progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| refresh      | Frequency for refreshing the progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| tidy_eval    | Logical, whether to enable tidy evaluation when interpreting command and pattern. If <code>TRUE</code> , you can use the "bang-bang" operator <code>!!</code> to programmatically insert the values of global objects.  |
| packages     | Character vector of packages to load right before the target builds or the output data is reloaded for downstream targets. Use <code>tar_option_set()</code> to set packages globally for all subsequent targets you define.  |
| library      | Character vector of library paths to try when loading packages.   |
| format       | Character of length 1, storage format of the data frames of posterior summaries and other data frames returned by targets. We recommend efficient data frame formats such as "feather" or "aws_parquet". For more on storage formats, see the help file of <code>targets::tar_target()</code> .   |
| format_df    | Character of length 1, storage format of the data frame targets such as posterior draws. We recommend efficient data frame formats such as "feather" or "aws_parquet". For more on storage formats, see the help file of <code>targets::tar_target()</code> .   |
| repository   | Character of length 1, remote repository for target storage. Choices: <ul style="list-style-type: none"> <li>"local": file system of the local machine.</li> <li>"aws": Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the <code>endpoint</code> argument of <code>tar_resources_aws()</code>, but versioning capabilities may be lost in doing so. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> <li>"gcp": Google Cloud Platform storage bucket. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> </ul> |

|                    |  |
|--------------------|--|
| error              | <p>Character of length 1, what to do if the target stops and throws an error. Options:</p> <ul style="list-style-type: none"> <li>• "stop": the whole pipeline stops and throws an error.</li> <li>• "continue": the whole pipeline keeps going.</li> <li>• "abridge": any currently running targets keep running, but no new targets launch after that. (Visit <a href="https://books.ropensci.org/targets/debugging.html">https://books.ropensci.org/targets/debugging.html</a> to learn how to debug targets using saved workspaces.)</li> <li>• "null": The errored target continues and returns NULL. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline.</li> </ul>  |
| memory             | <p>Character of length 1, memory strategy. If "persistent", the target stays in memory until the end of the pipeline (unless storage is "worker", in which case targets unloads the value from memory right after storing it in order to avoid sending copious data over a network). If "transient", the target gets unloaded after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value. For cloud-based dynamic files such as format = "aws_file", this memory strategy applies to temporary local copies of the file in <code>_targets/scratch/</code>: "persistent" means they remain until the end of the pipeline, and "transient" means they get deleted from the file system as soon as possible. The former conserves bandwidth, and the latter conserves local storage.</p>  |
| garbage_collection | <p>Logical, whether to run <code>base::gc()</code> just before the target runs.</p>  |
| deployment         | <p>Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code>. If "worker", the target builds on a parallel worker. If "main", the target builds on the host machine / process managing the pipeline.</p>  |
| priority           | <p>Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get built earlier (and polled earlier in <code>tar_make_future()</code>).</p>  |
| resources          | <p>Object returned by <code>tar_resources()</code> with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See <code>tar_resources()</code> for details.</p>  |
| storage            | <p>Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code>. Must be one of the following values:</p> <ul style="list-style-type: none"> <li>• "main": the target's return value is sent back to the host machine and saved/uploaded locally.</li> <li>• "worker": the worker saves/uploads the value.</li> <li>• "none": almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language. If you do use it, then the return value of the target is totally ignored when the target ends, but each downstream target still attempts to load the data file (except when <code>retrieval = "none"</code>).</li> </ul> <p>If you select <code>storage = "none"</code>, then the return value of the target's command is ignored, and the data is not saved automatically. As with dynamic files (format = "file" or "aws_file") it is the responsibility of the user to write to <code>tar_path()</code> from inside the target. An example target could</p> |

look something like `tar_target(x, saveRDS("value", tar_path(create_dir = TRUE)); "ignored", storage = "none")`.

The distinguishing feature of `storage = "none"` (as opposed to `format = "file"` or `"aws_file"`) is that in the general case, downstream targets will automatically try to load the data from the data store as a dependency. As a corollary, `storage = "none"` is completely unnecessary if `format` is `"file"` or `"aws_file"`.

|           |  |
|-----------|--|
| retrieval | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . Must be one of the following values: <ul style="list-style-type: none"> <li>• <code>"main"</code>: the target's dependencies are loaded on the host machine and sent to the worker before the target builds.</li> <li>• <code>"worker"</code>: the worker loads the targets dependencies.</li> <li>• <code>"none"</code>: the dependencies are not loaded at all. This choice is almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language.</li> </ul> |
| cue       | An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.  |

## Details

The MCMC targets use `R2jags::jags()` if `n.cluster` is 1 and `R2jags::jags.parallel()` otherwise. Most arguments to `tar_jags()` are forwarded to these functions.

## Value

`tar_jags_rep_draws()` returns list of target objects. See the "Target objects" section for background. The target names use the `name` argument as a prefix, and the individual elements of `jags_files` appear in the suffixes where applicable. As an example, the specific target objects returned by `tar_jags_rep_dic(name = x, jags_files = "y.jags")` are as follows.

- `x_file_y`: reproducibly track the JAGS model file. Returns a character vector of length 1 with the path to the JAGS model file.
- `x_lines_y`: read the contents of the JAGS model file for safe transport to parallel workers. Returns a character vector of lines in the model file.
- `x_data`: use dynamic branching to generate multiple JAGS datasets from the R expression in the `data` argument. Each dynamic branch returns a batch of JAGS data lists.
- `x_y`: run JAGS on each dataset from `x_data`. Each dynamic branch returns a tidy data frame of draws for each batch of data.
- `x`: combine all the batches from `x_y` into a non-dynamic target. Suppressed if `combine` is `FALSE`. Returns a long tidy data frame with all draws from all the branches of `x_y`.

## Target objects

Most `stantargets` functions are target factories, which means they return target objects or lists of target objects. Target objects represent skippable steps of the analysis pipeline as described at <https://books.ropensci.org/targets/>. Please read the walkthrough at <https://books.ropensci.org/targets/>.

[ropensci.org/targets/walkthrough.html](https://ropensci.org/targets/walkthrough.html) to understand the role of target objects in analysis pipelines.

For developers, <https://wlandau.github.io/targetopia/contributing.html#target-factories> explains target factories (functions like this one which generate targets) and the design specification at <https://books.ropensci.org/targets-design/> details the structure and composition of target objects.

## Examples

```
if (requireNamespace("R2jags", quietly = TRUE)) {
  targets::tar_dir({ # tar_dir() runs code from a temporary directory.
    targets::tar_script({
      library(jagstargets)
      # Do not use a temp file for a real project
      # or else your targets will always rerun.
      tmp <- tempfile(pattern = "", fileext = ".jags")
      tar_jags_example_file(tmp)
      list(
        tar_jags_rep_draws(
          your_model,
          jags_files = tmp,
          data = tar_jags_example_data(),
          parameters.to.save = "beta",
          batches = 2,
          reps = 2,
          stdout = R.utils::nullfile(),
          stderr = R.utils::nullfile()
        )
      )
    }, ask = FALSE)
  targets::tar_make()
})
}
```

---

tar\_jags\_rep\_summary *Tidy posterior summaries from multiple MCMCs per model*

---

## Description

Run multiple MCMCs on simulated datasets and return posterior summaries and the effective number of parameters for each run.

## Usage

```
tar_jags_rep_summary(
  name,
  jags_files,
  parameters.to.save,
  data = list(),
```

```

variables = NULL,
summaries = NULL,
summary_args = NULL,
batches = 1L,
reps = 1L,
combine = TRUE,
n.cluster = 1,
n.chains = 3,
n.iter = 2000,
n.burnin = as.integer(n.iter/2),
n.thin = 1,
jags.module = c("glm", "dic"),
inits = NULL,
RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry", "Super-Duper",
  "Mersenne-Twister"),
jags.seed = 123,
stdout = NULL,
stderr = NULL,
progress.bar = "text",
refresh = 0,
tidy_eval = targets::tar_option_get("tidy_eval"),
packages = targets::tar_option_get("packages"),
library = targets::tar_option_get("library"),
format = "qs",
format_df = "fst_tbl",
repository = targets::tar_option_get("repository"),
error = targets::tar_option_get("error"),
memory = "transient",
garbage_collection = targets::tar_option_get("garbage_collection"),
deployment = targets::tar_option_get("deployment"),
priority = targets::tar_option_get("priority"),
resources = targets::tar_option_get("resources"),
storage = targets::tar_option_get("storage"),
retrieval = targets::tar_option_get("retrieval"),
cue = targets::tar_option_get("cue")
)

```

## Arguments

|                    |   |
|--------------------|---|
| name               | Symbol, base name for the collection of targets. Serves as a prefix for target names.   |
| jags_files         | Character vector of JAGS model files. If you supply multiple files, each model will run on the one shared dataset generated by the code in data. If you supply an unnamed vector, <code>tools::file_path_sans_ext(basename(jags_files))</code> will be used as target name suffixes. If <code>jags_files</code> is a named vector, the suffixed will come from <code>names(jags_files)</code> . |
| parameters.to.save | Model parameters to save, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> .   |

|                           |  |
|---------------------------|--|
|                           | See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| <code>data</code>         | Code to generate the data list for the JAGS model. Optionally include a <code>.join_data</code> element to join parts of the data to correspondingly named parameters in the summary output. See the vignettes for details.                                  |
| <code>variables</code>    | Character vector of model parameter names. The output posterior summaries are restricted to these variables.   |
| <code>summaries</code>    | List of summary functions passed to <code>...</code> in <code>posterior::summarize_draws()</code> through <code>\$summary()</code> on the <code>CmdStanFit</code> object.  |
| <code>summary_args</code> | List of summary function arguments passed to <code>.args</code> in <code>posterior::summarize_draws()</code> through <code>\$summary()</code> on the <code>CmdStanFit</code> object.   |
| <code>batches</code>      | Number of batches. Each batch runs a model <code>reps</code> times.  |
| <code>reps</code>         | Number of replications per batch. Ideally, each rep should produce its own random dataset using the code supplied to <code>data</code> .   |
| <code>combine</code>      | Logical, whether to create a target to combine all the model results into a single data frame downstream. Convenient, but duplicates data.   |
| <code>n.cluster</code>    | Number of parallel processes, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.             |
| <code>n.chains</code>     | Number of MCMC chains, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                    |
| <code>n.iter</code>       | Number if iterations (including warmup), passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| <code>n.burnin</code>     | Number of warmup iterations, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.              |
| <code>n.thin</code>       | Thinning interval, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                        |
| <code>jags.module</code>  | Character vector of JAGS modules to load, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details. |
| <code>inits</code>        | Initial values of model parameters, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.       |
| <code>RNGname</code>      | Choice of random number generator, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.        |
| <code>jags.seed</code>    | Seeds to apply to JAGS, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.                   |

|              |   |
|--------------|---|
| stdout       | Character of length 1, file path to write the stdout stream of the model when it runs. Set to NULL to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stdout. Does not apply to messages, warnings, or errors.  |
| stderr       | Character of length 1, file path to write the stderr stream of the model when it runs. Set to NULL to print to the console. Set to <code>R.utils::nullfile()</code> to suppress stderr. Does not apply to messages, warnings, or errors.  |
| progress.bar | Type of progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.  |
| refresh      | Frequency for refreshing the progress bar, passed to <code>R2jags::jags()</code> or <code>R2jags::jags.parallel()</code> . See the argument documentation of the <code>R2jags::jags()</code> and <code>R2jags::jags.parallel()</code> help files for details.   |
| tidy_eval    | Logical, whether to enable tidy evaluation when interpreting command and pattern. If TRUE, you can use the "bang-bang" operator <code>!!</code> to programmatically insert the values of global objects.  |
| packages     | Character vector of packages to load right before the target builds or the output data is reloaded for downstream targets. Use <code>tar_option_set()</code> to set packages globally for all subsequent targets you define.  |
| library      | Character vector of library paths to try when loading packages.   |
| format       | Character of length 1, storage format of the data frames of posterior summaries and other data frames returned by targets. We recommend efficient data frame formats such as "feather" or "aws_parquet". For more on storage formats, see the help file of <code>targets::tar_target()</code> .   |
| format_df    | Character of length 1, storage format of the data frame targets such as posterior draws. We recommend efficient data frame formats such as "feather" or "aws_parquet". For more on storage formats, see the help file of <code>targets::tar_target()</code> .   |
| repository   | Character of length 1, remote repository for target storage. Choices: <ul style="list-style-type: none"> <li>• "local": file system of the local machine.</li> <li>• "aws": Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the <code>endpoint</code> argument of <code>tar_resources_aws()</code>, but versioning capabilities may be lost in doing so. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> <li>• "gcp": Google Cloud Platform storage bucket. See the cloud storage section of <a href="https://books.ropensci.org/targets/data.html">https://books.ropensci.org/targets/data.html</a> for details for instructions.</li> </ul> |
| error        | Character of length 1, what to do if the target stops and throws an error. Options: <ul style="list-style-type: none"> <li>• "stop": the whole pipeline stops and throws an error.</li> <li>• "continue": the whole pipeline keeps going.</li> <li>• "abridge": any currently running targets keep running, but no new targets launch after that. (Visit <a href="https://books.ropensci.org/targets/debugging.html">https://books.ropensci.org/targets/debugging.html</a> to learn how to debug targets using saved workspaces.)</li> <li>• "null": The errored target continues and returns NULL. The data hash is deliberately wrong so the target is not up to date for the next run of the pipeline.</li> </ul>  |



|                    |   |
|--------------------|---|
| memory             | Character of length 1, memory strategy. If "persistent", the target stays in memory until the end of the pipeline (unless storage is "worker", in which case targets unloads the value from memory right after storing it in order to avoid sending copious data over a network). If "transient", the target gets unloaded after every new target completes. Either way, the target gets automatically loaded into memory whenever another target needs the value. For cloud-based dynamic files such as format = "aws_file", this memory strategy applies to temporary local copies of the file in <code>_targets/scratch/</code> : "persistent" means they remain until the end of the pipeline, and "transient" means they get deleted from the file system as soon as possible. The former conserves bandwidth, and the latter conserves local storage.   |
| garbage_collection | Logical, whether to run <code>base::gc()</code> just before the target runs.  |
| deployment         | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . If "worker", the target builds on a parallel worker. If "main", the target builds on the host machine / process managing the pipeline.   |
| priority           | Numeric of length 1 between 0 and 1. Controls which targets get deployed first when multiple competing targets are ready simultaneously. Targets with priorities closer to 1 get built earlier (and polled earlier in <code>tar_make_future()</code> ).   |
| resources          | Object returned by <code>tar_resources()</code> with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See <code>tar_resources()</code> for details.  |
| storage            | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . Must be one of the following values: <ul style="list-style-type: none"> <li>"main": the target's return value is sent back to the host machine and saved/uploaded locally.</li> <li>"worker": the worker saves/uploads the value.</li> <li>"none": almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language. If you do use it, then the return value of the target is totally ignored when the target ends, but each downstream target still attempts to load the data file (except when <code>retrieval = "none"</code>).</li> </ul> <p>If you select <code>storage = "none"</code>, then the return value of the target's command is ignored, and the data is not saved automatically. As with dynamic files (<code>format = "file"</code> or <code>"aws_file"</code>) it is the responsibility of the user to write to <code>tar_path()</code> from inside the target. An example target could look something like <code>tar_target(x, saveRDS("value", tar_path(create_dir = TRUE))); "ignored", storage = "none")</code>.</p> <p>The distinguishing feature of <code>storage = "none"</code> (as opposed to <code>format = "file"</code> or <code>"aws_file"</code>) is that in the general case, downstream targets will automatically try to load the data from the data store as a dependency. As a corollary, <code>storage = "none"</code> is completely unnecessary if <code>format</code> is <code>"file"</code> or <code>"aws_file"</code>.</p> |
| retrieval          | Character of length 1, only relevant to <code>tar_make_clustermq()</code> and <code>tar_make_future()</code> . Must be one of the following values: <ul style="list-style-type: none"> <li>"main": the target's dependencies are loaded on the host machine and sent to the worker before the target builds.</li> </ul>   |

- "worker": the worker loads the targets dependencies.
  - "none": the dependencies are not loaded at all. This choice is almost never recommended. It is only for niche situations, e.g. the data needs to be loaded explicitly from another language.
- cue            An optional object from tar\_cue() to customize the rules that decide whether the target is up to date.

## Details

The MCMC targets use `R2jags::jags()` if `n.cluster` is 1 and `R2jags::jags.parallel()` otherwise. Most arguments to `tar_jags()` are forwarded to these functions.

## Value

`tar_jags_rep_summary()` returns list of target objects. See the "Target objects" section for background. The target names use the name argument as a prefix, and the individual elements of `jags_files` appear in the suffixes where applicable. As an example, the specific target objects returned by `tar_jags_rep_dic(name = x, jags_files = "y.jags")` are as follows.

- `x_file_y`: reproducibly track the JAGS model file. Returns a character vector of length 1 with the path to the JAGS model file.
- `x_lines_y`: read the contents of the JAGS model file for safe transport to parallel workers. Returns a character vector of lines in the model file.
- `x_data`: use dynamic branching to generate multiple JAGS datasets from the R expression in the data argument. Each dynamic branch returns a batch of JAGS data lists.
- `x_y`: run JAGS on each dataset from `x_data`. Each dynamic branch returns a tidy data frame of summaries for each batch of data.
- `x`: combine all the batches from `x_y` into a non-dynamic target. Suppressed if `combine` is FALSE. Returns a long tidy data frame with all summaries from all the branches of `x_y`.

## Target objects

Most `stantargets` functions are target factories, which means they return target objects or lists of target objects. Target objects represent skippable steps of the analysis pipeline as described at <https://books.ropensci.org/targets/>. Please read the walkthrough at <https://books.ropensci.org/targets/walkthrough.html> to understand the role of target objects in analysis pipelines.

For developers, <https://wlandau.github.io/targetopia/contributing.html#target-factories> explains target factories (functions like this one which generate targets) and the design specification at <https://books.ropensci.org/targets-design/> details the structure and composition of target objects.

## Examples

```
if (requireNamespace("R2jags", quietly = TRUE)) {
  targets::tar_dir({ # tar_dir() runs code from a temporary directory.
    targets::tar_script({
      library(jagstargets)
    })
  })
}
```

```
# Do not use a temp file for a real project
# or else your targets will always rerun.
tmp <- tempfile(pattern = "", fileext = ".jags")
tar_jags_example_file(tmp)
list(
  tar_jags_rep_summary(
    your_model,
    jags_files = tmp,
    data = tar_jags_example_data(),
    parameters.to.save = "beta",
    batches = 2,
    reps = 2,
    stdout = R.utils::nullfile(),
    stderr = R.utils::nullfile()
  )
), ask = FALSE)
targets::tar_make()
})
}
```

# Index

jagstargets (jagstargets-package), [2](#)  
jagstargets-package, [2](#)

tar\_jags, [3](#)  
tar\_jags(), [2](#)  
tar\_jags\_example\_data, [8](#)  
tar\_jags\_example\_file, [10](#)  
tar\_jags\_example\_file(), [9](#)  
tar\_jags\_rep\_dic, [10](#)  
tar\_jags\_rep\_draws, [16](#)  
tar\_jags\_rep\_summary, [21](#)  
tar\_jags\_rep\_summary(), [9](#)  
tar\_make\_clustermq(), [6](#), [7](#), [13](#), [14](#), [19](#), [20](#),  
[25](#)  
tar\_make\_future(), [6](#), [7](#), [13](#), [14](#), [19](#), [20](#), [25](#)  
tar\_path(), [7](#), [14](#), [19](#), [25](#)  
tar\_resources\_aws(), [6](#), [13](#), [18](#), [24](#)