

# Package ‘jagstargets’

November 18, 2024

**Title** Targets for JAGS Pipelines

**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** 1.2.2

**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), fst (>= 0.9.2), posterior (>= 1.0.1), purrr

(>= 0.3.4), qs2, R2jags (>= 0.6.1), rjags (>= 4.10), rlang (>=

0.4.10), secretbase (>= 0.4.0), stats, targets (>= 1.6.0),

tarchetypes (>= 0.8.0), tibble (>= 3.0.1), tidyselect, tools,

utils, withr (>= 2.1.2),

**Suggests** dplyr (>= 1.0.2), fs (>= 1.5.0), knitr (>= 1.30), qs (>=

0.23.2), 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** 7.3.2**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.oss@gmail.com>**Repository** CRAN**Date/Publication** 2024-11-18 14:50:02 UTC

## Contents

jagstargets-package . . . . .	2
tar_jags . . . . .	3
tar_jags_example_data . . . . .	9
tar_jags_example_file . . . . .	10
tar_jags_rep_dic . . . . .	11
tar_jags_rep_draws . . . . .	17
tar_jags_rep_summary . . . . .	23
<b>Index</b>	<b>31</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	<i>One MCMC per model with multiple outputs</i>
----------	---

---

## 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"),
```

```

    description = targets::tar_option_get("description")
  )

```

## Arguments

<code>name</code>	Symbol, base name for the collection of targets. Serves as a prefix for target names.
<code>jags_files</code>	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 <code>data</code> . 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> .
<code>parameters.to.save</code>	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>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>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 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.
<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.

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 runs 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>

- "aws": Amazon Web Services (AWS) S3 bucket. Can be configured with a non-AWS S3 bucket using the endpoint argument of `tar_resources_aws()`, but versioning capabilities may be lost in doing so. See the cloud storage section of <https://books.ropensci.org/targets/data.html> for details for instructions.
- "gcp": Google Cloud Platform storage bucket. See the cloud storage section of <https://books.ropensci.org/targets/data.html> for details for instructions.
- A character string from `tar_repository_cas()` for content-addressable storage.

Note: if repository is not "local" and format is "file" then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. The local file is deleted after the target runs.

error

Character of length 1, what to do if the target stops and throws an error. Options:

- "stop": the whole pipeline stops and throws an error.
- "continue": the whole pipeline keeps going.
- "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. In addition, as of version 1.8.0.9011, a value of NULL is given to upstream dependencies with `error = "null"` if loading fails.
- "abridge": any currently running targets keep running, but no new targets launch after that.
- "trim": all currently running targets stay running. A queued target is allowed to start if:
  1. It is not downstream of the error, and
  2. It is not a sibling branch from the same `tar_target()` call (if the error happened in a dynamic branch).

The idea is to avoid starting any new work that the immediate error impacts. `error = "trim"` is just like `error = "abridge"`, but it allows potentially healthy regions of the dependency graph to begin running. (Visit <https://books.ropensci.org/targets/debugging.html> to learn how to debug targets using saved workspaces.)

memory

Character of length 1, memory strategy. Possible values:

- "auto": new in targets version 1.8.0.9011, `memory = "auto"` is equivalent to `memory = "transient"` for dynamic branching (a non-null pattern argument) and `memory = "persistent"` for targets that do not use dynamic branching.
- "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).
- "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 (e.g. `format = "file"` with `repository = "aws"`), the `memory` option applies to the temporary local copy of the file: `"persistent"` means it remains until the end of the pipeline and is then deleted, and `"transient"` means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

<code>garbage_collection</code>	Logical: TRUE to run <code>base::gc()</code> just before the target runs, FALSE to omit garbage collection. In the case of high-performance computing, <code>gc()</code> runs both locally and on the parallel worker. All this garbage collection is skipped if the actual target is skipped in the pipeline. Non-logical values of <code>garbage_collection</code> are converted to TRUE or FALSE using <code>isTRUE()</code> . In other words, non-logical values are converted FALSE. For example, <code>garbage_collection = 2</code> is equivalent to <code>garbage_collection = FALSE</code> .
<code>deployment</code>	Character of length 1. If <code>deployment</code> is <code>"main"</code> , then the target will run on the central controlling R process. Otherwise, if <code>deployment</code> is <code>"worker"</code> and you set up the pipeline with distributed/parallel computing, then the target runs on a parallel worker. For more on distributed/parallel computing in targets, please visit <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> .
<code>priority</code>	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 dispatched earlier (and polled earlier in <code>tar_make_future()</code> ).
<code>resources</code>	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.
<code>storage</code>	Character string to control when the output of the target is saved to storage. Only relevant when using targets with parallel workers ( <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> ). Must be one of the following values: <ul style="list-style-type: none"> <li><code>"main"</code>: the target's return value is sent back to the host machine and saved/uploaded locally.</li> <li><code>"worker"</code>: the worker saves/uploads the value.</li> <li><code>"none"</code>: targets makes no attempt to save the result of the target to storage in the location where targets expects it to be. Saving to storage is the responsibility of the user. Use with caution.</li> </ul>
<code>retrieval</code>	Character string to control when the current target loads its dependencies into memory before running. (Here, a "dependency" is another target upstream that the current one depends on.) Only relevant when using targets with parallel workers ( <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> ). 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 runs.</li> <li><code>"worker"</code>: the worker loads the target's dependencies.</li> <li><code>"none"</code>: targets makes no attempt to load its dependencies. With <code>retrieval = "none"</code>, loading dependencies is the responsibility of the user. Use with caution.</li> </ul>
<code>cue</code>	An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.

**description** Character of length 1, a custom free-form human-readable text description of the target. Descriptions appear as target labels in functions like `tar_manifest()` and `tar_visnetwork()`, and they let you select subsets of targets for the `names` argument of functions like `tar_make()`. For example, `tar_manifest(names = tar_described_as(starts_with("survival model")))` lists all the targets whose descriptions start with the character string "survival model".

### 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 (identical(Sys.getenv("TAR_JAGS_EXAMPLES"), "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	<i>Tidy DIC output from multiple MCMCs per model</i>
------------------	--

---

### 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 = NULL,
  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"),
  description = targets::tar_option_get("description")
)
```

)

**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	The <code>jags.seed</code> argument of the <code>tar_jags_rep*</code> ( <code>)</code> functions is deprecated. See the "Seeds" section 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 runs 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> <li>• A character string from <code>tar_repository_cas()</code> for content-addressable storage.</li> </ul>

Note: if repository is not "local" and format is "file" then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. The local file is deleted after the target runs.

error

Character of length 1, what to do if the target stops and throws an error. Options:

- "stop": the whole pipeline stops and throws an error.
- "continue": the whole pipeline keeps going.
- "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. In addition, as of version 1.8.0.9011, a value of NULL is given to upstream dependencies with error = "null" if loading fails.
- "abridge": any currently running targets keep running, but no new targets launch after that.
- "trim": all currently running targets stay running. A queued target is allowed to start if:
  1. It is not downstream of the error, and
  2. It is not a sibling branch from the same `tar_target()` call (if the error happened in a dynamic branch).

The idea is to avoid starting any new work that the immediate error impacts. error = "trim" is just like error = "abridge", but it allows potentially healthy regions of the dependency graph to begin running. (Visit <https://books.ropensci.org/targets/debugging.html> to learn how to debug targets using saved workspaces.)

memory

Character of length 1, memory strategy. Possible values:

- "auto": new in targets version 1.8.0.9011, memory = "auto" is equivalent to memory = "transient" for dynamic branching (a non-null pattern argument) and memory = "persistent" for targets that do not use dynamic branching.
- "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).
- "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 (e.g. format = "file" with repository = "aws"), the memory option applies to the temporary local copy of the file: "persistent" means it remains until the end of the pipeline and is then deleted, and "transient" means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

garbage\_collection

Logical: TRUE to run `base::gc()` just before the target runs, FALSE to omit garbage collection. In the case of high-performance computing, `gc()` runs both locally and on the parallel worker. All this garbage collection is skipped if the actual target is skipped in the pipeline. Non-logical values of `garbage_collection`

are converted to TRUE or FALSE using `isTRUE()`. In other words, non-logical values are converted FALSE. For example, `garbage_collection = 2` is equivalent to `garbage_collection = FALSE`.

deployment	Character of length 1. If deployment is "main", then the target will run on the central controlling R process. Otherwise, if deployment is "worker" and you set up the pipeline with distributed/parallel computing, then the target runs on a parallel worker. For more on distributed/parallel computing in targets, please visit <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> .
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 dispatched 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 string to control when the output of the target is saved to storage. Only relevant when using targets with parallel workers ( <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> ). 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": targets makes no attempt to save the result of the target to storage in the location where targets expects it to be. Saving to storage is the responsibility of the user. Use with caution.</li> </ul>
retrieval	Character string to control when the current target loads its dependencies into memory before running. (Here, a "dependency" is another target upstream that the current one depends on.) Only relevant when using targets with parallel workers ( <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> ). 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 runs.</li> <li>• "worker": the worker loads the target's dependencies.</li> <li>• "none": targets makes no attempt to load its dependencies. With <code>retrieval = "none"</code>, loading dependencies is the responsibility of the user. Use with caution.</li> </ul>
cue	An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.
description	Character of length 1, a custom free-form human-readable text description of the target. Descriptions appear as target labels in functions like <code>tar_manifest()</code> and <code>tar_visnetwork()</code> , and they let you select subsets of targets for the names argument of functions like <code>tar_make()</code> . For example, <code>tar_manifest(names = tar_described_as(starts_with("survival model")))</code> lists all the targets whose descriptions start with the character string "survival model".

## 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.

**Seeds**

Rep-specific random number generator seeds for the data and models are automatically set based on the batch, rep, parent target name, and tar\_option\_get("seed"). This ensures the rep-specific seeds do not change when you change the batching configuration (e.g. 40 batches of 10 reps each vs 20 batches of 20 reps each). Each data seed is in the .seed list element of the output, and each JAGS seed is in the .seed column of each JAGS model output.

**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 (identical(Sys.getenv("TAR_JAGS_EXAMPLES"), "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)
    })
  })
}
```



```

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	<i>Tidy posterior draws from multiple MCMCs per model</i>
--------------------	---

---

### 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,
  transform = NULL,
  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 = NULL,
  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"),
description = targets::tar_option_get("description")
)

```

## 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> .
transform	Symbol or NULL, name of a function that accepts arguments <code>data</code> and <code>draws</code> and returns a data frame. Here, <code>data</code> is the JAGS data list supplied to the model, and <code>draws</code> is a data frame with one column per model parameter and one row per posterior sample. Any arguments other than <code>data</code> and <code>draws</code> must have valid default values because <code>jagstargets</code> will not populate them. See the simulation-based calibration discussion thread at <a href="https://github.com/ropensci/jagstargets/discussions/31">https://github.com/ropensci/jagstargets/discussions/31</a> for an example.
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	The <code>jags.seed</code> argument of the <code>tar_jags_rep*</code> ( <code>)</code> functions is deprecated. See the "Seeds" section 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 runs 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	<p>Character of length 1, remote repository for target storage. Choices:</p> <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 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> <li>• A character string from <code>tar_repository_cas()</code> for content-addressable storage.</li> </ul> <p>Note: if repository is not "local" and format is "file" then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. The local file is deleted after the target runs.</p>
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>• "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. In addition, as of version 1.8.0.9011, a value of NULL is given to upstream dependencies with <code>error = "null"</code> if loading fails.</li> <li>• "abridge": any currently running targets keep running, but no new targets launch after that.</li> <li>• "trim": all currently running targets stay running. A queued target is allowed to start if: <ol style="list-style-type: none"> <li>1. It is not downstream of the error, and</li> <li>2. It is not a sibling branch from the same <code>tar_target()</code> call (if the error happened in a dynamic branch).</li> </ol> <p>The idea is to avoid starting any new work that the immediate error impacts. <code>error = "trim"</code> is just like <code>error = "abridge"</code>, but it allows potentially healthy regions of the dependency graph to begin running. (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.)</p> </li> </ul>
memory	Character of length 1, memory strategy. Possible values:

- "auto": new in targets version 1.8.0.9011, memory = "auto" is equivalent to memory = "transient" for dynamic branching (a non-null pattern argument) and memory = "persistent" for targets that do not use dynamic branching.
- "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).
- "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 (e.g. format = "file" with repository = "aws"), the memory option applies to the temporary local copy of the file: "persistent" means it remains until the end of the pipeline and is then deleted, and "transient" means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

#### garbage\_collection

Logical: TRUE to run `base::gc()` just before the target runs, FALSE to omit garbage collection. In the case of high-performance computing, `gc()` runs both locally and on the parallel worker. All this garbage collection is skipped if the actual target is skipped in the pipeline. Non-logical values of `garbage_collection` are converted to TRUE or FALSE using `isTRUE()`. In other words, non-logical values are converted FALSE. For example, `garbage_collection = 2` is equivalent to `garbage_collection = FALSE`.

#### deployment

Character of length 1. If deployment is "main", then the target will run on the central controlling R process. Otherwise, if deployment is "worker" and you set up the pipeline with distributed/parallel computing, then the target runs on a parallel worker. For more on distributed/parallel computing in targets, please visit <https://books.ropensci.org/targets/crew.html>.

#### 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 dispatched earlier (and polled earlier in `tar_make_future()`).

#### resources

Object returned by `tar_resources()` with optional settings for high-performance computing functionality, alternative data storage formats, and other optional capabilities of targets. See `tar_resources()` for details.

#### storage

Character string to control when the output of the target is saved to storage. Only relevant when using targets with parallel workers (<https://books.ropensci.org/targets/crew.html>). Must be one of the following values:

- "main": the target's return value is sent back to the host machine and saved/uploaded locally.
- "worker": the worker saves/uploads the value.
- "none": targets makes no attempt to save the result of the target to storage in the location where targets expects it to be. Saving to storage is the responsibility of the user. Use with caution.

#### retrieval

Character string to control when the current target loads its dependencies into memory before running. (Here, a "dependency" is another target upstream that

the current one depends on.) Only relevant when using targets with parallel workers (<https://books.ropensci.org/targets/crew.html>). Must be one of the following values:

- "main": the target's dependencies are loaded on the host machine and sent to the worker before the target runs.
- "worker": the worker loads the target's dependencies.
- "none": targets makes no attempt to load its dependencies. With retrieval = "none", loading dependencies is the responsibility of the user. Use with caution.

cue	An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.
description	Character of length 1, a custom free-form human-readable text description of the target. Descriptions appear as target labels in functions like <code>tar_manifest()</code> and <code>tar_visnetwork()</code> , and they let you select subsets of targets for the names argument of functions like <code>tar_make()</code> . For example, <code>tar_manifest(names = tar_described_as(starts_with("survival model")))</code> lists all the targets whose descriptions start with the character string "survival model".

## 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`.

## Seeds

Rep-specific random number generator seeds for the data and models are automatically set based on the batch, rep, parent target name, and `tar_option_get("seed")`. This ensures the rep-specific seeds do not change when you change the batching configuration (e.g. 40 batches of 10 reps each vs 20 batches of 20 reps each). Each data seed is in the `.seed` list element of the output, and each JAGS seed is in the `.seed` column of each JAGS model output.

## 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 (identical(Sys.getenv("TAR_JAGS_EXAMPLES"), "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 = NULL,
  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"),
  description = targets::tar_option_get("description")
)

```

**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, `tools::file_path_sans_ext(basename(jags_files))` will be used as target name suffixes. If `jags_files` is a named vector, the suffixed will come from `names(jags_files)`.

<code>parameters.to.save</code>	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 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.
<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.

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	The <code>jags.seed</code> argument of the <code>tar_jags_rep*</code> ( <code>)</code> functions is deprecated. See the "Seeds" section 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 runs 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> <li>• A character string from <code>tar_repository_cas()</code> for content-addressable storage.</li> </ul>

Note: if repository is not "local" and format is "file" then the target should create a single output file. That output file is uploaded to the cloud and tracked for changes where it exists in the cloud. The local file is deleted after the target runs.

`error` Character of length 1, what to do if the target stops and throws an error. Options:

- "stop": the whole pipeline stops and throws an error.
- "continue": the whole pipeline keeps going.
- "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. In addition, as of version 1.8.0.9011, a value of NULL is given to upstream dependencies with `error = "null"` if loading fails.
- "abridge": any currently running targets keep running, but no new targets launch after that.
- "trim": all currently running targets stay running. A queued target is allowed to start if:
  1. It is not downstream of the error, and
  2. It is not a sibling branch from the same `tar_target()` call (if the error happened in a dynamic branch).

The idea is to avoid starting any new work that the immediate error impacts. `error = "trim"` is just like `error = "abridge"`, but it allows potentially healthy regions of the dependency graph to begin running. (Visit <https://books.ropensci.org/targets/debugging.html> to learn how to debug targets using saved workspaces.)

`memory` Character of length 1, memory strategy. Possible values:

- "auto": new in targets version 1.8.0.9011, `memory = "auto"` is equivalent to `memory = "transient"` for dynamic branching (a non-null pattern argument) and `memory = "persistent"` for targets that do not use dynamic branching.
- "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).
- "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 (e.g. `format = "file"` with `repository = "aws"`), the memory option applies to the temporary local copy of the file: "persistent" means it remains until the end of the pipeline and is then deleted, and "transient" means it gets deleted as soon as possible. The former conserves bandwidth, and the latter conserves local storage.

`garbage_collection`

Logical: TRUE to run `base::gc()` just before the target runs, FALSE to omit garbage collection. In the case of high-performance computing, `gc()` runs both locally and on the parallel worker. All this garbage collection is skipped if the actual target is skipped in the pipeline. Non-logical values of `garbage_collection`

are converted to TRUE or FALSE using `isTRUE()`. In other words, non-logical values are converted FALSE. For example, `garbage_collection = 2` is equivalent to `garbage_collection = FALSE`.

deployment	Character of length 1. If deployment is "main", then the target will run on the central controlling R process. Otherwise, if deployment is "worker" and you set up the pipeline with distributed/parallel computing, then the target runs on a parallel worker. For more on distributed/parallel computing in targets, please visit <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> .
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 dispatched 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 string to control when the output of the target is saved to storage. Only relevant when using targets with parallel workers ( <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> ). 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": targets makes no attempt to save the result of the target to storage in the location where targets expects it to be. Saving to storage is the responsibility of the user. Use with caution.</li> </ul>
retrieval	Character string to control when the current target loads its dependencies into memory before running. (Here, a "dependency" is another target upstream that the current one depends on.) Only relevant when using targets with parallel workers ( <a href="https://books.ropensci.org/targets/crew.html">https://books.ropensci.org/targets/crew.html</a> ). 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 runs.</li> <li>• "worker": the worker loads the target's dependencies.</li> <li>• "none": targets makes no attempt to load its dependencies. With <code>retrieval = "none"</code>, loading dependencies is the responsibility of the user. Use with caution.</li> </ul>
cue	An optional object from <code>tar_cue()</code> to customize the rules that decide whether the target is up to date.
description	Character of length 1, a custom free-form human-readable text description of the target. Descriptions appear as target labels in functions like <code>tar_manifest()</code> and <code>tar_visnetwork()</code> , and they let you select subsets of targets for the names argument of functions like <code>tar_make()</code> . For example, <code>tar_manifest(names = tar_described_as(starts_with("survival model")))</code> lists all the targets whose descriptions start with the character string "survival model".

## 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.

**Seeds**

Rep-specific random number generator seeds for the data and models are automatically set based on the batch, rep, parent target name, and tar\_option\_get("seed"). This ensures the rep-specific seeds do not change when you change the batching configuration (e.g. 40 batches of 10 reps each vs 20 batches of 20 reps each). Each data seed is in the .seed list element of the output, and each JAGS seed is in the .seed column of each JAGS model output.

**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 (identical(Sys.getenv("TAR_JAGS_EXAMPLES"), "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)
    })
  })
}
```

```
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, 9  
tar\_jags\_example\_file, 10  
tar\_jags\_example\_file(), 9, 10  
tar\_jags\_rep\_dic, 11  
tar\_jags\_rep\_draws, 17  
tar\_jags\_rep\_summary, 23  
tar\_jags\_rep\_summary(), 10  
tar\_make(), 8, 15, 22, 28  
tar\_make\_future(), 7, 15, 21, 28  
tar\_manifest(), 8, 15, 22, 28  
tar\_repository\_cas(), 6, 13, 20, 26  
tar\_resources\_aws(), 6, 13, 20, 26  
tar\_target(), 6, 14, 20, 27  
tar\_visnetwork(), 8, 15, 22, 28