# Package: chkptstanr (via r-universe)

October 25, 2024

**Title** Checkpoint MCMC Sampling with 'Stan'

**Version** 0.2.0

**Description** Fit Bayesian models in Stan <doi:10.18637/jss.v076.i01> with checkpointing, that is, the ability to stop the MCMC sampler at will, and then pick right back up where the MCMC sampler left off. Custom 'Stan' models can be fitted, or the popular package 'brms' <doi:10.18637/jss.v080.i01> can be used to generate the 'Stan' code. This package is fully compatible with the R packages 'brms', 'posterior', 'cmdstanr', and 'bayesplot'.

**License** Apache License 2.0 | file LICENSE

**Depends** R (>= 4.1.0)

**Imports** brms (>= 2.16.1), abind, methods, rstan, Rdpack, fs, waldo, glue, stringr

**Suggests** cmdstanr, rmarkdown, knitr, posterior, here, testthat (>= 3.0.0), withr, ape

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**Additional_repositories** `https://mc-stan.org/r-packages/`

**RdMacros** Rdpack

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**URL** `https://github.com/venpopov/chkptstanr`, `https://venpopov.github.io/chkptstanr/`

**BugReports** `https://github.com/venpopov/chkptstanr/issues`

**Repository** https://popov-lab.r-universe.dev

**RemoteUrl** https://github.com/venpopov/chkptstanr

**RemoteRef** v0.2.0

**RemoteSha** 94eea9a665608323c9ee080914cdf06be3502766

## Contents

---

`chkpt_brms` *Checkpoint Sampling: brms*

---

### Description

Fit Bayesian generalized (non-)linear multivariate multilevel models using brms with checkpointing.

### Usage

```
chkpt_brms(
  formula,
  data,
  iter_adaptation = 150,
  iter_warmup = 1000,
  iter_sampling = 1000,
  iter_per_chkpt = 100,
  parallel_chains = 4,
  threads_per = 1,
  chkpt_progress = TRUE,
  control = NULL,
  seed = 1,
  stop_after = NULL,
  reset = FALSE,
  path,
  ...
)
```

## Arguments

| | |
|---|---|
| `formula` | An object of class `formula`, `brmsformula`, or `brms`{`mvbrmsformula`}. Further information can be found in `brmsformula`. |
| `data` | An object of class `data.frame` (or one that can be coerced to that class) containing data of all variables used in the model. |
| `iter_adaptation` | (positive integer) The number of iterations in the initial warmup, which are used for the adaptation of the step size and inverse mass matrix. This is equivalent to the traditional warmup stage. Checkpointing will begin only after this stage is complete. |
| `iter_warmup` | (positive integer) The number of warmup iterations to run per chain after the adaptation stage (defaults to 1000). During this stage the step size and inverse mass matrix are fixed to the values found during the adaptation stage. There is no further adaptation performed. |
| `iter_sampling` | (positive integer) The number of post-warmup iterations to run per chain (defaults to 1000). |
| `iter_per_chkpt` | (positive integer). The number of iterations per checkpoint. Note that `iter_sampling` is divided by `iter_per_chkpt` to determine the number of checkpoints. This must result in an integer (if not, there will be an error). |
| `parallel_chains` | (positive integer) The *maximum number* of MCMC chains to run in parallel. If parallel_chains is not specified then the default is to look for the option `mc.cores`, which can be set for an entire R session by `options(mc.cores=value)`. If the `mc.cores` option has not been set then the default is `1`. |
| `threads_per` | (positive integer) Number of threads to use in within-chain parallelization (defaults to `1`). |
| `chkpt_progress` | logical. Should the `chkptstanr` progress be printed (defaults to `TRUE`) ? If set to `FALSE`, the standard `cmdstanr` progress bar is printed for each checkpoint (which does not actually keep track of checkpointing progress) |
| `control` | A named list of parameters to control the sampler's behavior. It defaults to NULL so all the default values are used. For a comprehensive overview see `stan`. |
| `seed` | (positive integer). The seed for random number generation to make results reproducible. |
| `stop_after` | (positive integer). The number of iterations to sample before stopping. If `NULL`, then all iterations are sampled (defaults to `NULL`). Note that sampling will stop at the end of the first checkpoint which has an iteration number greater than or equal to `stop_after`. |
| `reset` | logical. Should the checkpointing be reset? If `TRUE`, then the model will begin sampling from the beginning (defaults to `FALSE`). WARNING: This will remove all previous checkpointing information (see `reset_checkpoints()`). If the model is unchanged and previously compiled, sampling will begin without recompiling the model. |

| | |
|---|---|
| `path` | Character string. The path to the folder, that is used for saving the checkpoints (see Details). You can provide either a relative path to the current working directory or a full path. You no longer need to create the folder, as this is done automatically. |
| `...` | Any additional arguments passed to `brm`, including, but not limited to, user-defined prior distributions, the `brmsfamily` (e.g., `family = poisson()`), data2, custom_families, etc. |

## Value

An object of class `brmsfit`

## Note

A folder specified by `path` is created with four subfolders:

- **cmd_output**: The cmdstanr output_files (one for each checkpoint and chain).
- **cp_info**: Mass matrix, step size, and initial values for next checkpoint (last iteration from previous checkpoint).
- **cp_samples**: Samples from the posterior distribution (post warmup)
- **stan_model**: Complied **Stan** model

## Examples

```
## Not run:
library(brms)
library(cmdstanr)


# "random" intercept
fit1 <- chkpt_brms(
  bf(
    formula = count ~ zAge + zBase * Trt + (1 | patient),
    family = poisson()
  ),
  data = epilepsy, ,
  iter_warmup = 1000,
  iter_sampling = 1000,
  iter_per_chkpt = 250,
  path = "chkpt_folder_fit1"
)

# brmsfit output
fit1


# remove "random" intercept (for model comparison)
fit2 <- chkpt_brms(
  bf(
    formula = count ~ zAge + zBase * Trt,
    family = poisson()
  ),
  data = epilepsy, ,
  iter_warmup = 1000,
  iter_sampling = 1000,
  iter_per_chkpt = 250,
  path = "chkpt_folder_fit2"
)

# brmsfit output
fit2

# compare models
loo(fit1, fit2)


# priors
bprior <- prior(constant(1), class = "b") +
  prior(constant(2), class = "b", coef = "zBase") +
  prior(constant(0.5), class = "sd")

# fit model
fit3 <-
  chkpt_brms(
    bf(
      formula = count ~ zAge + zBase + (1 | patient),
      family = poisson()
    ),
    data = epilepsy,
    path = "chkpt_folder_fit3",
    prior = bprior,
    iter_warmup = 1000,
    iter_sampling = 1000,
    iter_per_chkpt = 250,
  )

# check priors
prior_summary(fit3)

## End(Not run)
```

## `chkpt_setup` *Checkpoint Setup*

### Description

Deterimine the number of checkpoints for the warmup and sampling, given the desired number of iterations for each and the iterations per checkpoint.

### Usage

```
chkpt_setup(iter_sampling, iter_warmup, iter_per_chkpt)
```

### Arguments

| | |
|---|---|
| `iter_sampling` | (positive integer) The number of post-warmup iterations to run per chain. Note: in the CmdStan User's Guide this is referred to as num_samples. |
| `iter_warmup` | (positive integer) The number of warmup iterations to run per chain. Note: in the CmdStan User's Guide this is referred to as num_warmup. |
| `iter_per_chkpt` | (positive integer) The number of iterations per check point. |

### Value

A list with the following:

- `warmup_chkpts`: Number of warmup checkpoints
- `sample_chkpts`: Number of sampling checkpoints
- `total_chkpts`: Total number of checkpoints (warmup_chkpts + sample_chkpts)
- `iter_per_chkpt`: Iterations per checkpoint

### Examples

```
chkpt_setup <- chkpt_setup(
  iter_sampling = 5000,
  iter_warmup = 2000,
  iter_per_chkpt = 10
)

chkpt_setup
```

---

## `chkpt_stan` *Checkpoint Sampling: Stan*

---

### Description

Fit Bayesian models using Stan with checkpointing.

### Usage

```
chkpt_stan(
  model_code,
  data,
  iter_adaptation = 150,
  iter_warmup = 1000,
  iter_sampling = 1000,
  iter_per_chkpt = 100,
  parallel_chains = 4,
  threads_per = 1,
  chkpt_progress = TRUE,
  control = NULL,
  seed = 1,
  stop_after = NULL,
  reset = FALSE,
  path,
  ...
)
```

### Arguments

| | |
|---|---|
| `model_code` | Character string corresponding to the Stan model. |
| `data` | A named list of R objects (like for RStan). Further details can be found in `sample`. |
| `iter_adaptation` | (positive integer) The number of iterations in the initial warmup, which are used for the adaptation of the step size and inverse mass matrix. This is equivalent to the traditional warmup stage. Checkpointing will begin only after this stage is complete. |
| `iter_warmup` | (positive integer) The number of warmup iterations to run per chain (defaults to 1000). |
| `iter_sampling` | (positive integer) The number of post-warmup iterations to run per chain (defaults to 1000). |
| `iter_per_chkpt` | (positive integer). The number of iterations per checkpoint. Note that `iter_sampling` is divided by `iter_per_chkpt` to determine the number of checkpoints. This must result in an integer (if not, there will be an error). |
| `parallel_chains` | (positive integer) The *maximum number* of MCMC chains to run in parallel. If parallel_chains is not specified then the default is to look for the option `mc.cores`, which can be set for an entire R session by `options(mc.cores=value)`. If the `mc.cores` option has not been set then the default is `1`. |
| `threads_per` | (positive integer) Number of threads to use in within-chain parallelization (defaults to `1`). |
| `chkpt_progress` | logical. Should the `chkptstanr` progress be printed (defaults to `TRUE`) ? If set to `FALSE`, the standard `cmdstanr` progress bar is printed for each checkpoint (which does not actually keep track of checkpointing progress) |
| `control` | A named list of parameters to control the sampler's behavior. It defaults to NULL so all the default values are used. For a comprehensive overview see `stan`. |
| `seed` | (positive integer). The seed for random number generation to make results reproducible. |
| `stop_after` | (positive integer). The number of iterations to sample before stopping. If `NULL`, then all iterations are sampled (defaults to `NULL`). Note that sampling will stop at the end of the first checkpoint which has an iteration number greater than or equal to `stop_after`. |

| | |
|---|---|
| `reset` | logical. Should the checkpointing be reset? If `TRUE`, then the model will begin sampling from the beginning (defaults to `FALSE`). WARNING: This will remove all previous checkpointing information (see `reset_checkpoints()`). If the model is unchanged and previously compiled, sampling will begin without recompiling the model. |
| `path` | Character string. The path to the folder, that is used for saving the checkpoints (see Details). You can provide either a relative path to the current working directory or a full path. You no longer need to create the folder, as this is done automatically. |
| `...` | Currently ignored. |

## Value

An objet of class `chkpt_stan`

## Examples

```
## Not run:

stan_code <- make_stancode(
  bf(
    formula = count ~ zAge + zBase * Trt + (1 | patient),
    family = poisson()
  ),
  data = epilepsy
)
stan_data <- make_standata(
  bf(
    formula = count ~ zAge + zBase * Trt + (1 | patient),
    family = poisson()
  ),
  data = epilepsy
)

# "random" intercept
fit1 <- chkpt_stan(
  model_code = stan_code,
  data = stan_data,
  iter_warmup = 1000,
  iter_sampling = 1000,
  iter_per_chkpt = 250,
  path = "chkpt_folder_fit1"
)

draws <- combine_chkpt_draws(object = fit1)

posterior::summarise_draws(draws)


# eight schools example
```

```
stan_code <- "
data {
 int<lower=0> n;
  real y[n];
  real<lower=0> sigma[n];
}
parameters {
  real mu;
  real<lower=0> tau;
  vector[n] eta;
}
transformed parameters {
  vector[n] theta;
  theta = mu + tau * eta;
}
model {
  target += normal_lpdf(eta | 0, 1);
  target += normal_lpdf(y | theta, sigma);
}
"
stan_data <- schools.data <- list(
  n = 8,
  y = c(28, 8, -3, 7, -1, 1, 18, 12),
  sigma = c(15, 10, 16, 11, 9, 11, 10, 18)
)

fit2 <- chkpt_stan(
  model_code = stan_code,
  data = stan_data,
  iter_warmup = 1000,
  iter_sampling = 1000,
  iter_per_chkpt = 250,
  path = "chkpt_folder_fit2"
)

draws <- combine_chkpt_draws(object = fit2)

posterior::summarise_draws(draws)

## End(Not run)
```

---

## `combine_chkpt_draws` *Combine Checkpoint Draws*

---

### Description

Combine Checkpoint Draws

### Usage

```
combine_chkpt_draws(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `brmsfit` or `chkpt_stan`. |
| `...` | Currently ignored. |

## Value

An object of class `draws_array`.

## Examples

```
## Not run:
path <- create_folder(folder_name = "chkpt_folder_fit1")

stan_code <- "
data {
 int<lower=0> n;
  real y[n];
  real<lower=0> sigma[n];
}
parameters {
  real mu;
  real<lower=0> tau;
  vector[n] eta;
}
transformed parameters {
  vector[n] theta;
  theta = mu + tau * eta;
}
model {
  target += normal_lpdf(eta | 0, 1);
  target += normal_lpdf(y | theta, sigma);
}
"

stan_data <- schools.data <- list(
  n = 8,
  y = c(28,  8, -3,  7, -1,  1, 18, 12),
  sigma = c(15, 10, 16, 11,  9, 11, 10, 18)
)

fit2 <- chkpt_stan(model_code = stan_code,
                   data = stan_data,
                   iter_warmup = 1000,
                   iter_sampling = 1000,
                   iter_per_chkpt = 250,
                   path = path)

draws <- combine_chkpt_draws(object = fit2)

draws

## End(Not run)
```

## `create_folder` *Create Folder for Checkpointing (Deprecated)*

### Description

create_folder() is deprecated. Provide a path directly in chkpt_brms()', or chkpt_stan() instead and a folder will be created automatically.'

### Usage

```
create_folder(folder_name = "cp_folder", path = NULL)
```

### Arguments

| | |
|---|---|
| `folder_name` | Character string. Desired name for the 'parent' folder (defaults to `cp_folder`). |
| `path` | Character string, when specified. Defaults to `NULL`, which then makes the folder in the working directory. |

### Value

the path to the main parent folder containing the four subfolders. This path should be used as the `path` argument in `chkpt_brms`. If `return_relative = TRUE`, the relative path to the current working directory is returned. If `path` is specified or `return_relative = FALSE`, the full path is returned.

### ————-

Create the folder for checkingpointing, which will 'house' additional folders for the `.stan` model, checkpointing information, and draws from the posterior distribution.

### Note

This creates a directory with four folders:

- **cmd_output**: The cmdstanr output_files (one for each checkpoint and chain).
- **cp_info**: Mass matrix, step size, and initial values for next checkpoint (last iteration from previous checkpoint).
- **cp_samples**: Samples from the posterior distribution (post warmup)
- **stan_model**: Complied **Stan** model

## Examples

```
# create initial folder
path <- create_folder(folder_name = 'cp_folder')
path
unlink('cp_folder', recursive = TRUE) # remove folder

# remove folder
unlink('cp_folder', recursive = TRUE)
identical(dir(path), character(0))

# repeat - no warning
path <- create_folder(folder_name = 'cp_folder')

# repeat - warning, but folders are kept
path <- create_folder(folder_name = 'cp_folder')
identical(dir(path), c('cmd_output', 'cp_info', 'cp_samples', 'stan_model'))

unlink('cp_folder', recursive = TRUE)

# specify nested folder
path <- create_folder(folder_name = 'nested_folder/cp_folder')
path
unlink('nested_folder', recursive = TRUE) # remove folder
```

---

`extract_chkpt_draws` *Extract Draws from* `CmdStanMCMC` *Objects*

---

## Description

A convenience function for extracting the draws from a `CmdStanMCMC` object.

## Usage

```
extract_chkpt_draws(object, phase)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `CmdStanMCMC`. |
| `phase` | Character string. Which phase during checkpointing? The options included `warmup` and `sample`. The latter extracts the draws with `inc_warmup = FALSE`, which is the default in `draws` |

## Value

A 3-D `draws_array` object (iteration *x* chain *x* variable).

## Note

This can be used to extract the draws in general by setting `phase = "sample"` which then only includes the post-warmup draws.

## Examples

```
## Not run:
 library(cmdstanr)

# eight schools example
fit_schools_ncp_mcmc <- cmdstanr_example("schools_ncp")

drws <- extract_chkpt_draws(object = fit_schools_ncp_mcmc,
                            phase = "sample")

# compare to cmdstanr
all.equal(drws, fit_schools_ncp_mcmc$draws())

## End(Not run)
```

---

`extract_hmc_info` *Extract HMC Sampler Information*

---

## Description

Extract the inverse metric and step size adaption from `CmdStanMCMC` objects.

## Usage

```
extract_hmc_info(object)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `CmdStanMCMC` |

## Value

A list including

- `inv_metric`: Inverse metric for each chain (with `matrix = FALSE`).
- `step_size_adapt`: Step size adaptation for each chain.

## Note

This is primarily used internally.

## Examples

```
## Not run:

library(cmdstanr)

fit_schools_ncp_mcmc <- cmdstanr_example("schools_ncp")
```

```
extract_hmc_info(fit_schools_ncp_mcmc)

## End(Not run)
```

## extract_stan_state *Extract Stan State*

### Description

Extract Stan State

### Usage

```
extract_stan_state(object, phase)
```

### Arguments

| | |
|---|---|
| object | An object of class cmdstanr |
| phase | Character string indicating the current phase. Options include wormup and sample/ |

### Value

A list containing the inverse metric, step size, and last MCMC draw (to be used as the initial value for the next checkpoint)

### Examples

```
## Not run:
library(cmdstanr)

# eight schools example
fit_schools_ncp_mcmc <- cmdstanr_example("schools_ncp")

extract_stan_state(fit_schools_ncp_mcmc, "sample")

## End(Not run)
```

## make_brmsfit — *Make* `brmsfit` *Object*

### Description

This is primarily used internally, wherein the output files of multiple cmdstanr fits are combined into a single `brmsfit` object. object is converted into a `brmsfit` object.

### Usage

```
make_brmsfit(formula, data, path, ...)
```

### Arguments

| | |
|---|---|
| `formula` | A `brms` formula used to generate the checkpoints |
| `data` | A data frame used to generate the checkpoints |
| `path` | Character string. The path to the folder, that is used for saving the checkpoints. |
| `...` | Additional arguments to be passed to `brm`. |

### Value

An object of class `brmsfit`

### Note

This is primarily an internal function that constructs a `brmsfit` object.

## print.chkpt_brms — *Print* `chkpt_brms` *Objects*

### Description

Print `chkpt_brms` Objects

### Usage

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

### Arguments

| | |
|---|---|
| `x` | Object of class `chkpt_brms` |
| `...` | Currently ignored |

### Value

No return value, and used to print the chkpt_brms object.

### Note

This function mainly avoids printing out a list, and it is only used when brmsfit = "FALSE" in chkpt_brms.

Typically, after fitting, the posterior draws should be summarized with combine_chkpt_draws (assuming brmsfit = "FALSE").

---

## print.chkpt_setup *Print* chkpt_setup *Object*

---

### Description

Print chkpt_setup Object

### Usage

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

### Arguments

| | |
|---|---|
| x | An object of class chkpt_setup. |
| ... | Currently ignored. |

### Value

No return value, and used to print the chkpt_setup object.

### Examples

```
chkpt_setup <- chkpt_setup(
  iter_sampling = 5000,
  iter_warmup = 2000,
  iter_per_chkpt = 10
)


chkpt_setup
```

## `reset_checkpoints` *Delete Checkpoint Folders containing samples, keep the model*

### Description

Deletes all checkpoint files and folders under path except for stan_model/model.stan and stan_model/model.exe. This allows you to restart the sampling from 0 without recompiling the model.

### Usage

```
reset_checkpoints(path, reset = TRUE, recompile = FALSE)
```

### Arguments

| | |
|---|---|
| `path` | (character) The path to the checkpoint folder. |
| `reset` | (logical) If TRUE, only the checkpoint folders are deleted |
| `recompile` | (logical) If TRUE, the entire folder is deleted allowing for a fresh start. If both `reset` and `recompile` are `FALSE`, nothing is done. |

# Index