# Package: drjacoby (via r-universe)

November 4, 2026

**Type** Package

**Title** Flexible Markov Chain Monte Carlo via Reparameterization

**Version** 1.5.4

**Description** drjacoby is an R package for performing Bayesian inference via Markov chain monte carlo (MCMC). In addition to being highly flexible it implements some advanced techniques that can improve mixing in tricky situations.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**LinkingTo** Rcpp

**Imports** Rcpp, coda, parallel, ggplot2, usethis, tools, rlang, cowplot, dplyr, magrittr, GGally, tidyr

**Suggests** testthat, covr, knitr, rmarkdown, microbenchmark, gridExtra

**BugReports** https://github.com/mrc-ide/drjacoby/issues

**VignetteBuilder** knitr

**Depends** R (>= 3.1.0)

**Config/pak/sysreqs** cmake git make libgit2-dev libicu-dev libuv1-dev libssl-dev libx11-dev

**Repository** https://ncov-ic.r-universe.dev

**Date/Publication** 2024-06-26 02:17:18 UTC

**RemoteUrl** https://github.com/mrc-ide/drjacoby

**RemoteRef** master

**RemoteSha** edfea6339eb5a828410a0109bc9fef70c22ee210

## Contents


acf_data . . . . . 2
check_drjacoby_loaded . . . . . 2

cpp_template . . . 3
define_params . . . 3
drjacoby . . . 4
gelman_rubin . . . 4
plot_autocorrelation . . . 5
plot_cor_mat . . . 5
plot_credible . . . 6
plot_density . . . 6
plot_mc_acceptance . . . 7
plot_pairs . . . 7
plot_rung_loglike . . . 8
plot_scatter . . . 8
plot_trace . . . 9
population_data . . . 10
run_mcmc . . . 10
sample_chains . . . 12

**Index** **13**

---

`acf_data` *Estimate autocorrelation*

---

## Description

Estimate autocorrelation

## Usage

```
acf_data(x, lag)
```

## Arguments

| | |
|---|---|
| `x` | Single chain. |
| `lag` | maximum lag. Must be an integer between 1 and 500. |

---

`check_drjacoby_loaded` *Check that drjacoby package has loaded successfully*

---

## Description

Simple function to check that drjacoby package has loaded successfully. Prints "drjacoby loaded successfully!" if so.

## Usage

```
check_drjacoby_loaded()
```

## cpp_template *Create template for cpp*

### Description

Create template for cpp

### Usage

```
cpp_template(save_as)
```

### Arguments

| | |
|---|---|
| save_as | Path of (.cpp) file to create, relative to root of active project. |

## define_params *Define parameters dataframe*

### Description

Provides a convenient way of defining parameters in the format required by `run_mcmc()`. Each parameter must have the following three elements, defined in order:

- `name` - the parameter name.
- `min` - the minimum value of the parameter. `-Inf` is allowed.
- `max` - the maximum value of the parameter. `Inf` is allowed.

There following arguments are also optional:

- `init` - the initial value of the parameter. If running multiple chains a vector of initial values can be used to specify distinct values for each chain.
- `block` - which likelihood block(s) this parameter belongs to. See vignettes for instructions on using likelihood blocks.

### Usage

```
define_params(...)
```

### Arguments

| | |
|---|---|
| ... | a series of named input arguments. |

## Examples

```
define_params(name = "mu", min = -10, max = 10, init = 0,
              name = "sigma", min = 0, max = 5, init = c(1, 2))

define_params(name = "mu1", min = -10, max = 10, init = 0, block = 1,
              name = "mu2", min = -10, max = 10, init = 0, block = 2,
              name = "sigma", min = 0, max = 5, init = 1, block = c(1, 2))
```

---

# drjacoby *Flexible Markov Chain Monte Carlo via Reparameterization*

---

## Description

Flexible Markov chain monte carlo via reparameterization using the Jacobean matrix.

_PACKAGE

---

# gelman_rubin *Gelman-Rubin statistic*

---

## Description

Estimate sthe Gelman-Rubin (rhat) convergence statistic for a single parameter across multiple chains. Basic method, assuming all chains are of equal length

## Usage

```
gelman_rubin(par_matrix, chains, samples)
```

## Arguments

| | |
|---|---|
| par_matrix | Matrix (interations x chains) |
| chains | number of chains |
| samples | number of samples |

## Value

Gelman-Rubin statistic

## References

Gelman, A., and D. B. Rubin. 1992. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science 7: 457–511.

https://astrostatistics.psu.edu/RLectures/diagnosticsMCMC.pdf

## plot_autocorrelation *Plot autocorrelation*

### Description

Plot autocorrelation for specified parameters

### Usage

```
plot_autocorrelation(x, lag = 20, par = NULL, chain = 1, phase = "sampling")
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| lag | maximum lag. Must be an integer between 1 and 500. |
| par | vector of parameter names. If NULL all parameters are plotted. |
| chain | which chain to plot. |
| phase | which phase to plot. Must be either "burnin" or "sampling". |

## plot_cor_mat *Plot posterior correlation matrix*

### Description

Produces a matrix showing the correlation between all parameters from posterior draws.

### Usage

```
plot_cor_mat(x, show = NULL, phase = "sampling", param_names = NULL)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| show | Vector of parameter names to plot. |
| phase | which phase to plot. Must be either "burnin" or "sampling". |
| param_names | Optional vector of names to replace the default parameter names. |

## plot_credible *Plot 95% credible intervals*

### Description

Plots posterior 95% credible intervals over specified set of parameters (defauls to all parameters).

### Usage

```
plot_credible(x, show = NULL, phase = "sampling", param_names = NULL)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| show | vector of parameter names to plot. |
| phase | which phase to plot. Must be either "burnin" or "sampling". |
| param_names | optional vector of names to replace the default parameter names. |

## plot_density *Produce density plots*

### Description

Density plots of all parameters. Use show and hide to be more specific about which parameters to plot.

### Usage

```
plot_density(x, show = NULL, hide = NULL)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| show | optional vector of parameter names to plot. Parameters matching show will be included. |
| hide | optional vector of parameter names to filter out. Parameters matching hide will be hidden. |

## plot_mc_acceptance *Plot Metropolis coupling acceptance rates*

### Description

Plot Metropolis coupling acceptance rates between all rungs.

### Usage

```
plot_mc_acceptance(x, chain = NULL, phase = "sampling", x_axis_type = 1)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| chain | which chain to plot. If NULL then plot all chains. |
| phase | which phase to plot. Must be either "burnin" or "sampling". |
| x_axis_type | how to format the x-axis. 1 = integer rungs, 2 = values of the thermodynamic power. |

## plot_pairs *Produce scatterplots between multiple parameters*

### Description

Uses ggpairs function from the GGally package to produce scatterplots between all named parameters.

### Usage

```
plot_pairs(x, show = NULL, hide = NULL)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| show | optional vector of parameter names to plot. Parameters matching show will be included. |
| hide | optional vector of parameter names to filter out. Parameters matching hide will be hidden. |

## plot_rung_loglike *Plot loglikelihood 95% credible intervals*

### Description

Plot loglikelihood 95% credible intervals.

### Usage

```
plot_rung_loglike(
  x,
  chain = 1,
  phase = "sampling",
  x_axis_type = 1,
  y_axis_type = 1
)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| chain | which chain to plot. |
| phase | which phase to plot. Must be either "burnin" or "sampling". |
| x_axis_type | how to format the x-axis. 1 = integer rungs, 2 = values of the thermodynamic power. |
| y_axis_type | how to format the y-axis. 1 = raw values, 2 = truncated at auto-chosen lower limit. 3 = double-log scale. |

## plot_scatter *Produce bivariate scatterplot*

### Description

Produces scatterplot between two named parameters.

### Usage

```
plot_scatter(
  x,
  parameter1,
  parameter2,
  downsample = TRUE,
  phase = "sampling",
  chain = NULL
)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| parameter1 | name of parameter first parameter. |
| parameter2 | name of parameter second parameter. |
| downsample | whether to downsample output to 200 values max to speed up plotting. |
| phase | which phase to plot. Must be either "burnin" or "sampling". |
| chain | which chain to plot. |

---

## plot_trace *Plot parameter trace*

---

### Description

Produce a series of plots corresponding to each parameter, including the raw trace, the posterior histogram and an autocorrelation plot. Plotting objects can be cycled through interactively, or can be returned as an object allowing them to be viewed/edited by the user.

### Usage

```
plot_trace(
  x,
  show = NULL,
  hide = NULL,
  lag = 20,
  downsample = TRUE,
  phase = "sampling",
  chain = NULL,
  display = TRUE
)
```

### Arguments

| | |
|---|---|
| x | an object of class drjacoby_output |
| show | optional vector of parameter names to plot. Parameters matching show will be included. |
| hide | optional vector of parameter names to filter out. Parameters matching hide will be hidden. |
| lag | maximum lag. Must be an integer between 1 and 500. |
| downsample | boolean. Whether to downsample chain to make plotting more efficient. |
| phase | which phase to plot. Must be either "burnin", "sampling" or "both". |
| chain | which chain to plot. |
| display | boolean. Whether to show plots, if FALSE then plotting objects are returned without displaying. |

## population_data *Population data.*

### Description

Example population growth data.

### Usage

```
population_data
```

### Format

A data frame with 20 rows and 2 variables:

**pop** population size

**time** time ...

## run_mcmc *Run drjacoby MCMC*

### Description

Run MCMC either with or without parallel tempering turned on. Minimum inputs include a data object, a data.frame of parameters, a log-likelihood function and a log-prior function. Produces an object of class `drjacoby_output`, which contains all MCMC output along with some diagnostics and a record of inputs.

### Usage

```
run_mcmc(
  data,
  df_params,
  misc = list(),
  loglike,
  logprior,
  burnin = 1000,
  samples = 10000,
  rungs = 1,
  chains = 5,
  beta_manual = NULL,
  alpha = 1,
  target_acceptance = 0.44,
  cluster = NULL,
  coupling_on = TRUE,
```

```
  pb_markdown = FALSE,
  save_data = TRUE,
  save_hot_draws = FALSE,
  silent = FALSE
)
```

## Arguments

| | |
|---|---|
| data | a named list or data frame or data values. |
| df_params | a data.frame of parameters (see ?define_params). |
| misc | optional list object passed to likelihood and prior. This can be useful for passing values that are not strictly data, for example passing a lookup table to the prior function. |
| loglike, logprior | the log-likelihood and log-prior functions used in the MCMC. Can either be passed in as R functions (not in quotes), or as character strings naming compiled C++ functions (in quotes). |
| burnin | the number of burn-in iterations. Automatic tuning of proposal standard deviations is only active during the burn-in period. |
| samples | the number of sampling iterations. |
| rungs | the number of temperature rungs used in the parallel tempering method. By default, $\beta$ values are equally spaced between 0 and 1, i.e. $\beta[i]$ =(i-1)/(rungs-1) for i in 1:rungs. The likelihood for the i<sup>th</sup> heated chain is raised to the power $\beta[i]^{\alpha}$, meaning we can use the $\alpha$ parameter to concentrate rungs towards the start or the end of the interval (see the alpha argument). |
| chains | the number of independent replicates of the MCMC to run. If a cluster object is defined then these chains are run in parallel, otherwise they are run in serial. |
| beta_manual | vector of manually defined $\beta$ values used in the parallel tempering approach. If defined, this overrides the spacing defined by rungs. Note that even manually defined $\beta$ values are raised to the power $\alpha$ internally, hence you should set alpha = 1 if you want to fix $\beta$ values exactly. |
| alpha | the likelihood for the i<sup>th</sup> heated chain is raised to the power $\beta[i]^{\alpha}$, meaning we can use the $\alpha$ parameter to concentrate rungs towards the start or the end of the temperature scale. |
| target_acceptance | Target acceptance rate. Should be between 0 and 1. Default of 0.44, set as optimum for unvariate proposal distributions. |
| cluster | option to pass in a cluster environment, allowing chains to be run in parallel (see package "parallel"). |
| coupling_on | whether to implement Metropolis-coupling over temperature rungs. The option of deactivating coupling has been retained for general interest and debugging purposes only. If this parameter is FALSE then parallel tempering will have no impact on MCMC mixing. |
| pb_markdown | whether to run progress bars in markdown mode, meaning they are only updated when they reach 100% to avoid large amounts of output being printed to markdown files. |

| | |
|---|---|
| save_data | if TRUE (the default) the raw input data is stored for reference in the project output. This allows complete reproducibility from a project, but may be undesirable when datasets are very large. |
| save_hot_draws | if TRUE the parameter draws relating to the hot chains are also stored inside the pt element of the project output. If FALSE (the default) only log-likelihoods and log-priors are stored from heated chains. |
| silent | whether to suppress all console output. |

## Details

Note that both data and misc are passed into log-likelihood/log-prior functions *by reference*. This means if you modify these objects inside the functions then any changes will persist.

---

# sample_chains *Sample posterior draws from all available chains*

---

## Description

Sample posterior draws from all available chains

## Usage

```
sample_chains(x, sample_n, keep_chain_index = FALSE)
```

## Arguments

| | |
|---|---|
| x | an object of class drjacoby_output. |
| sample_n | An integer number of samples. |
| keep_chain_index | if TRUE then the column giving the chain is retained. |

## Value

A data.frame of posterior samples

# Index

∗ **datasets**
population_data, 10

acf_data, 2

check_drjacoby_loaded, 2
cpp_template, 3

define_params, 3
drjacoby, 4

gelman_rubin, 4

plot_autocorrelation, 5
plot_cor_mat, 5
plot_credible, 6
plot_density, 6
plot_mc_acceptance, 7
plot_pairs, 7
plot_rung_loglike, 8
plot_scatter, 8
plot_trace, 9
population_data, 10

run_mcmc, 10

sample_chains, 12