

# Package ‘armspp’

July 22, 2025

**Title** Adaptive Rejection Metropolis Sampling (ARMS) via 'Rcpp'

**Version** 0.0.2

**Description** An efficient 'Rcpp' implementation of the Adaptive Rejection Metropolis Sampling (ARMS) algorithm proposed by Gilks, W. R., Best, N. G. and Tan, K. K. C. (1995) <doi:10.2307/2986138>. This allows for sampling from a univariate target probability distribution specified by its (potentially unnormalised) log density.

**Depends** R (>= 3.2.3)

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**LinkingTo** Rcpp

**Imports** Rcpp

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown, covr, testthat

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Michael Bertolacci [aut, cre]

**Maintainer** Michael Bertolacci <m.bertolacci@gmail.com>

**Repository** CRAN

**Date/Publication** 2019-05-24 09:20:03 UTC

## Contents

|                      |          |
|----------------------|----------|
| arms . . . . .       | 2        |
| arms_gibbs . . . . . | 4        |
| <b>Index</b>         | <b>6</b> |

**Description**

This function performs Adaptive Rejection Metropolis Sampling to sample from a target distribution specified by its (potentially unnormalised) log density. The function constructs a rejection distribution based on piecewise linear functions that envelop the log density of the target.

If the target is log-concave, the `metropolis` parameter can be set to `FALSE`, and an accept-reject sampling scheme is used which yields independent samples.

Otherwise, if `metropolis` is `TRUE`, a Metropolis-Hastings step is used to construct a Markov chain with a stationary distribution matching the target. It is possible in this case for the rejection distribution to be a poor proposal, so users should be careful to check the output matches the desired distribution.

All arguments other than `n_samples`, `include_n_evaluations` and `arguments` can be either vectors or lists as appropriate. If they are vectors, they will be recycled in the same manner as, e.g., `rnorm`. The entries of arguments may be vectors/lists and will also be recycled (see examples).

**Usage**

```
arms(n_samples, log_pdf, lower, upper, previous = (upper + lower)/2,
     initial = lower + (1:n_initial) * (upper - lower)/(n_initial + 1),
     n_initial = 10, convex = 0, max_points = max(2 * n_initial + 1,
     100), metropolis = TRUE, include_n_evaluations = FALSE,
     arguments = list())
```

**Arguments**

|                                    |                                                                                                                              |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>n_samples</code>             | Number of samples to return.                                                                                                 |
| <code>log_pdf</code>               | Potentially unnormalised log density of target distribution. Can also be a list of functions.                                |
| <code>lower</code>                 | Lower bound of the support of the target distribution.                                                                       |
| <code>upper</code>                 | Upper bound of the support of the target distribution.                                                                       |
| <code>previous</code>              | The previous value of the Markov chain to be used if <code>metropolis = TRUE</code> .                                        |
| <code>initial</code>               | Initial points with which to build the rejection distribution.                                                               |
| <code>n_initial</code>             | Number of points used to form <code>initial</code> ; ignored if <code>initial</code> provided.                               |
| <code>convex</code>                | Convexity adjustment.                                                                                                        |
| <code>max_points</code>            | Maximum number of points to allow in the rejection distribution.                                                             |
| <code>metropolis</code>            | Whether to use a Metropolis-Hastings step after rejection sampling. Not necessary if the target distribution is log concave. |
| <code>include_n_evaluations</code> | Whether to return an object specifying the number of function evaluations used.                                              |
| <code>arguments</code>             | List of additional arguments to be passed to <code>log_pdf</code>                                                            |

**Value**

Vector or matrix of samples if `include_n_evaluations` is `FALSE`, otherwise a list.

**References**

Gilks, W. R., Best, N. G. and Tan, K. K. C. (1995) Adaptive rejection Metropolis sampling. *Applied Statistics*, 44, 455-472.

**See Also**

[http://www1.maths.leeds.ac.uk/~wally.gilks/adaptive.rejection/web\\_page/Welcome.html](http://www1.maths.leeds.ac.uk/~wally.gilks/adaptive.rejection/web_page/Welcome.html)

**Examples**

```
# The normal distribution, which is log concave, so metropolis can be FALSE
result <- arms(
  1000, dnorm, -1000, 1000, metropolis = FALSE,
  arguments = list(log = TRUE), include_n_evaluations = TRUE
)
print(result$n_evaluations)
hist(result$samples, freq = FALSE, br = 20)
curve(dnorm(x), min(result$samples), max(result$samples), col = 'red', add = TRUE)

# Mixture of normals: 0.4 N(-1, 1) + 0.6 N(4, 1). Not log concave.
dnormmixture <- function(x) {
  parts <- log(c(0.4, 0.6)) + dnorm(x, mean = c(-1, 4), log = TRUE)
  log(sum(exp(parts - max(parts)))) + max(parts)
}
samples <- arms(1000, dnormmixture, -1000, 1000)
hist(samples, freq = FALSE)

# List of log pdfs, demonstrating recycling of log_pdf argument
samples <- arms(
  10,
  list(
    function(x) -x ^ 2 / 2,
    function(x) -(x - 10) ^ 2 / 2
  ),
  -1000,
  1000
)
print(samples)

# Another way to achieve the above, this time with recycling in arguments
samples <- arms(
  10, dnorm, -1000, 1000,
  arguments = list(
    mean = c(0, 10), sd = 1, log = TRUE
  )
)
print(samples)
```

arms\_gibbs

*Gibbs sampling using ARMS***Description**

This function uses ARMS (see also [arms](#)) to sample from a multivariate target distribution specified by its (potentially unnormalised) log density using Gibbs sampling. The function updates each argument to the log pdf in turn using ARMS, returning a matrix of samples.

The arguments to this function have the same meaning as for [arms](#), except here they are recycled along the dimension of previous, rather than from sample to sample.

**Usage**

```
arms_gibbs(n_samples, previous, log_pdf, lower, upper, initial = NULL,
           n_initial = 10, convex = 0, max_points = 100, metropolis = TRUE,
           include_n_evaluations = FALSE, show_progress = FALSE)
```

**Arguments**

|                       |                                                                                                                              |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------|
| n_samples             | Number of samples to return.                                                                                                 |
| previous              | Starting value for the Gibbs sampler. Vectors of this length are passed to log_pdf as the first argument.                    |
| log_pdf               | Potentially unnormalised log density of target distribution.                                                                 |
| lower                 | Lower bound of the support of the target distribution.                                                                       |
| upper                 | Upper bound of the support of the target distribution.                                                                       |
| initial               | Initial points with which to build the rejection distribution.                                                               |
| n_initial             | Number of points used to form initial; ignored if initial provided.                                                          |
| convex                | Convexity adjustment.                                                                                                        |
| max_points            | Maximum number of points to allow in the rejection distribution.                                                             |
| metropolis            | Whether to use a Metropolis-Hastings step after rejection sampling. Not necessary if the target distribution is log concave. |
| include_n_evaluations | Whether to return an object specifying the number of function evaluations used.                                              |
| show_progress         | If TRUE, a progress bar is shown.                                                                                            |

**Value**

Matrix of samples if include\_n\_evaluations is FALSE, otherwise a list.

**References**

Gilks, W. R., Best, N. G. and Tan, K. K. C. (1995) Adaptive rejection Metropolis sampling. *Applied Statistics*, 44, 455-472.

**See Also**

[http://www1.maths.leeds.ac.uk/~wally.gilks/adaptive.rejection/web\\_page/Welcome.html](http://www1.maths.leeds.ac.uk/~wally.gilks/adaptive.rejection/web_page/Welcome.html)

**Examples**

```
# The classic 8schools example from RStan
# https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started
schools_data <- list(
  J = 8,
  y = c(28, 8, -3, 7, -1, 1, 18, 12),
  sigma = c(15, 10, 16, 11, 9, 11, 10, 18)
)

log_pdf <- function(params, p) {
  mu <- params[1]
  tau <- params[2]
  eta <- params[3 : (3 + schools_data$J - 1)]
  output <- (
    sum(dnorm(eta, 0, 1, log = TRUE)) +
    sum(dnorm(
      schools_data$y,
      mu + tau * eta,
      schools_data$sigma,
      log = TRUE
    ))
  )
  return(output)
}

x_start <- c(0, 0, rep(0, schools_data$J))
names(x_start) <- c(
  'mu',
  'tau',
  paste0('eta', 1 : schools_data$J)
)

samples <- arms_gibbs(
  250,
  x_start,
  log_pdf,
  c(-1000, 0, rep(-1000, schools_data$J)),
  1000,
  metropolis = FALSE
)
print(colMeans(samples[51 : 250, ]))
```

# Index

arms, [2](#), [4](#)  
arms\_gibbs, [4](#)