Package 'Rtwalk'

October 30, 2026

Type Package

Title An MCMC Sampler Using the t-Walk Algorithm

Version 2.2.0

Maintainer Rodrigo Fonseca Villa <rodrigo03.villa@gmail.com>

Description Implements the t-walk algorithm, a general-purpose, self-adjusting Markov Chain Monte Carlo (MCMC) sampler for continuous distributions as described by Christen & Fox (2010) <doi:10.1214/10-BA603>. The t-walk requires no tuning and is robust for a wide range of target distributions, including high-dimensional and multimodal problems. The sampler is implemented in C++ and validated against a pure R reference implementation of the published algorithm. The package supports reproducible parallel chains, continuation of runs, single transitions for hybrid samplers, and convergence diagnostics following Vehtari et al. (2021) <doi:10.1214/20-BA1221>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

Depends R (>= 3.6.0)

Imports codetools, parallel, stats, utils, graphics, grDevices, coda, Rcpp

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), covr, mvtnorm, devtools, pkgload, posterior, roxygen2, knitr, rmarkdown, ellipse

VignetteBuilder knitr

URL https://github.com/rodrigosqrt3/Rtwalk

BugReports https://github.com/rodrigosqrt3/Rtwalk/issues

NeedsCompilation yes

Author Rodrigo Fonseca Villa [aut, cre] (ORCID: <https://orcid.org/0009-0005-2938-2270>)

Repository CRAN

Date/Publication 2026-10-30 09:51:03 UTC

Contents

calculate_diagnostics 2
coda-methods 3
plot.twalk 4
posterior-methods 5
twalk 6
twalk-methods 9
twalk_continue 10
twalk_control 11
twalk_step 13
visualize_results 14

Index **16**

`calculate_diagnostics` *Calculate MCMC diagnostics*

Description

Computes posterior summaries, effective sample sizes, Monte Carlo standard errors (MCSE), integrated autocorrelation times (IAT), and R-hat values after discarding a burn-in fraction.

Usage

```
calculate_diagnostics(
  samples,
  burnin_frac = 0.2,
  param_names = NULL,
  title = ""
)
```

Arguments

`samples`	A matrix of MCMC samples (iterations x parameters), or a list of matrices for multiple independent chains. For a list, burn-in is discarded separately from each chain.
`burnin_frac`	Fraction of samples to discard as burn-in.
`param_names`	Optional character vector of parameter names. By default, the column names of `samples` are used, if any.
`title`	Optional title for printed output.

Details

The convergence diagnostics follow Vehtari et al. (2021) and agree with those of the **posterior** package:

- `ESS` is the effective sample size for the mean, estimated from split chains with Geyer's (1992) initial monotone sequence; `MCSE = SD / sqrt(ESS)` and `IAT = N / ESS`, where `N` is the number of retained draws. The IAT is the integrated autocorrelation time used by Christen & Fox (2010) to compare samplers.
- `ESS_bulk` and `ESS_tail` are the rank-normalized bulk and tail effective sample sizes.
- `Rhat` is the rank-normalized split-R-hat (the maximum of its bulk and tail versions). Because chains are split in halves, it is also defined for a single chain. Values above 1.01 indicate lack of convergence.

When chains have different lengths, ESS and R-hat use the first draws of each chain up to the length of the shortest one.

Value

A data frame with posterior summaries, ESS (bulk and tail), MCSE, IAT, and R-hat, returned invisibly after being printed.

References

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B. and Buerkner, P.-C. (2021). Rank-normalization, folding, and localization: An improved R-hat for assessing convergence of MCMC. *Bayesian Analysis*, 16(2), 667–718. doi:10.1214/20BA1221

Geyer, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science*, 7(4), 473–483.

Examples

```
log_post <- function(x) dnorm(x, log = TRUE)
res <- twalk(log_post, n_iter = 2000, x0 = -2, xp0 = 2)
calculate_diagnostics(
  res$samples,
  burnin_frac = 0.2,
  param_names = "theta",
  title = "Standard normal"
)
```

`coda-methods` *Convert t-walk output to coda objects*

Description

Converts the primary or companion trajectory from a `twalk` result into the standard MCMC classes supplied by the `coda` package.

Usage

```
## S3 method for class 'twalk'
as.mcmc(x, burnin_frac = 0, companion = FALSE, ...)

## S3 method for class 'twalk'
as.mcmc.list(x, burnin_frac = 0, companion = FALSE, ...)
```

Arguments

x	An object of class twalk.
burnin_frac	Fraction of iterations to discard from the beginning of each chain.
companion	Logical; if TRUE, convert the companion trajectory instead of the primary trajectory.
...	Additional arguments reserved for compatibility with the coda generics.

Details

as.mcmc() is intended for single-chain results. For parallel results, use as.mcmc.list() so that chain boundaries remain available to convergence diagnostics.

Value

as.mcmc() returns a coda::mcmc object. as.mcmc.list() returns a coda::mcmc.list object.

Examples

```
log_post <- function(x) -0.5 * sum(x^2)
set.seed(1)
fit <- twalk(log_post, n_iter = 2000, x0 = c(-1, 1), xp0 = c(1, -1),
             show_progress = FALSE)
chain <- coda::as.mcmc(fit, burnin_frac = 0.2)
coda::effectiveSize(chain)
```

---

plot.twalk *Plot t-walk output*

Description

Draws trace plots of the log posterior and of selected parameters, with one colour per chain, next to their marginal densities. The trace of the log posterior is the most useful single summary to assess burn-in.

Usage

```
## S3 method for class 'twalk'
plot(x, pars = NULL, burnin_frac = 0, ...)
```

Arguments

x	An object of class twalk.
pars	Indices or names of the parameters to plot. Defaults to the first (up to) four parameters.
burnin_frac	Fraction of each chain to omit from the plots.
...	Unused.

Value

x, invisibly.

Examples

```
log_post <- function(x) -0.5 * sum(x^2)
fit <- twalk(log_post, n_iter = 2000, x0 = c(a = -1, b = 1),
             xp0 = c(1, -1), show_progress = FALSE)
plot(fit)
```

---

posterior-methods *Convert t-walk output to posterior draws objects*

Description

Methods for the generics of the **posterior** package, which give access to its summaries, diagnostics and plotting ecosystem (for example `posterior::summarise_draws()` or the **bayesplot** package). Chain boundaries are preserved, and the log posterior of the stored points is included as the variable `lp__`, following the convention of Stan.

Usage

```
## S3 method for class 'twalk'
as_draws_array(x, burnin_frac = 0, companion = FALSE, ...)

## S3 method for class 'twalk'
as_draws(x, burnin_frac = 0, companion = FALSE, ...)

## S3 method for class 'twalk'
as_draws_df(x, burnin_frac = 0, companion = FALSE, ...)
```

Arguments

x	An object of class twalk.
burnin_frac	Fraction of iterations to discard from the beginning of each chain.
companion	Logical; if TRUE, convert the companion trajectory instead of the primary one.
...	Unused.

Details

These methods are registered when **posterior** is loaded; the package is only suggested.

Value

A `draws_array` (or `draws_df`) object.

Examples

```
if (requireNamespace("posterior", quietly = TRUE)) {
  log_post <- function(x) -0.5 * sum(x^2)
  fit <- twalk(log_post, n_iter = 2000, x0 = c(a = -1, b = 1),
               xp0 = c(1, -1), show_progress = FALSE)
  draws <- posterior::as_draws_array(fit, burnin_frac = 0.2)
  posterior::summarise_draws(draws)
}
```

`twalk` *Run the t-walk MCMC Algorithm*

Description

This function implements the t-walk algorithm by Christen & Fox (2010), a general-purpose MCMC sampler that does not require manual tuning. The function can run multiple independent MCMC chains in parallel to accelerate execution and facilitate convergence diagnostics.

Usage

```
twalk(
  log_posterior,
  n_iter,
  x0,
  xp0,
  n_chains = 1,
  n_cores = NULL,
  show_progress = TRUE,
  thin = 1,
  control = twalk_control(),
  ...
)
```

Arguments

`log_posterior`	A function that takes a parameter vector as its first argument and returns one numeric log posterior density. It may return `-Inf` outside the support, but must not return vectors, `NA`, `NaN`, or `+Inf`. If `x0` has names, the parameter vector passed to `log_posterior` carries the same names. Additional arguments can be passed to this function via `...`.

n_iter	The number of iterations to run for each chain.
x0	A numeric vector with the initial values for the first point (x), or a matrix with one row per chain.
xp0	A numeric vector with the initial values for the second point (x'), or a matrix with one row per chain. It must differ from x0; identical initial points produce a degenerate t-walk state.
n_chains	The number of independent MCMC chains to run. Defaults to 1, which runs a single chain sequentially. If greater than 1, parallel mode is activated. Convergence diagnostics such as R-hat are most informative when the chains start from dispersed points, supplied as matrices in x0 and xp0.
n_cores	The number of CPU cores to use in parallel mode. If NULL (default), it will attempt to use all available cores minus one. Each chain receives its own random-number stream derived from R's current random state, so calling set.seed() before twalk() makes results reproducible, and the results do not depend on n_cores.
show_progress	Logical; whether to display progress bars and status messages. Defaults to TRUE.
thin	Positive integer; only every thin-th iteration is stored. All iterations are still run.
control	A list of algorithm constants created by twalk_control(). The defaults are those recommended by Christen & Fox (2010).
...	Additional arguments to be passed to the log_posterior function.

Details

The t-walk maintains two points, x and x', and is a Metropolis–Hastings sampler on the product space with target $\pi(x)\pi(x')$. At each iteration one of four moves (Traverse, Walk, Blow, Hop) is chosen and only one of the two points is updated. The sampler is invariant to changes of scale and location of the target, so no tuning is required. The kernel is implemented in C++; twalk_control() gives access to the constants of the algorithm and to a pure-R reference implementation.

Value

An object of class twalk, a list containing:

samples	The primary t-walk trajectory, with n_iter / thin rows per chain. Columns are named after x0, if it has names.
companion_samples	The auxiliary trajectory maintained by the t-walk, or NULL if store_companion = FALSE in twalk_control().
log_posterior, companion_log_posterior	The log posterior evaluated at the stored points of each trajectory, useful to monitor burn-in.
acceptance_rate	The average Metropolis–Hastings acceptance rate across all chains. Accepted identity proposals are included, as required by the t-walk transition kernel.
move_rate	The proportion of iterations in which an accepted proposal actually changed at least one of the two t-walk points.

`no_move_rate` The proportion of iterations containing an accepted identity proposal. It equals `acceptance_rate - move_rate`.

`kernel_statistics`
A data frame reporting the number of proposals, acceptances, actual movements, acceptance rate, and movement rate for each of the Traverse, Walk, Blow, and Hop kernels.

`n_iter` The number of iterations run per chain.

`thin` The thinning interval.

`n_chains` The number of independent chains.

`total_iterations`
The total number of iterations run over all chains (`n_iter * n_chains`).

`n_dim` The dimension of the parameter space.

`n_evaluations` The number of successful log-posterior evaluations.

`n_errors`, `first_error`
The number of evaluations that raised an error and the first error message.

`final_state` For single-chain results, the final points and their log posterior values, used by `twalk_continue()`.

`control` The `twalk_control()` settings used.

`individual_chains`
If `n_chains > 1`, a list containing the raw results from each separate chain, useful for diagnostics like R-hat.

The field `all_samples` of earlier versions is deprecated: it is no longer stored, and accessing it computes it on the fly with a warning.

Errors in `log_posterior`

Return `-Inf` outside the support of the target. If `log_posterior` raises an error at a proposal, the proposal is rejected (as if outside the support) and a single warning summarizing the errors is issued at the end.

Interrupting

In sequential mode, interrupting the computation (for example with Esc or Ctrl+C) returns the iterations completed so far, with a warning.

References

Christen, J. A. and Fox, C. (2010). A general purpose sampling algorithm for continuous distributions (the t-walk). *Bayesian Analysis*, 5(2), 263–282. doi:10.1214/10BA603

See Also

`twalk_continue()` to extend a run, `twalk_step()` to use the t-walk kernel inside other samplers, `summary.twalk()`, `plot.twalk()`.

Examples

```
# Example 1: Sampling from a Bivariate Normal (sequential mode)
# The 'mvtnorm' package is required for this example
if (requireNamespace("mvtnorm", quietly = TRUE)) {
  log_post <- function(x) {
   mvtnorm::dmvnorm(x, mean = c(0, 0), sigma = matrix(c(1, 0.8, 0.8, 1), 2, 2), log = TRUE)
  }

  # Set a seed for reproducibility
  set.seed(123)
  result_seq <- twalk(log_posterior = log_post, n_iter = 5000,
                      x0 = c(-1, 1), xp0 = c(1, -1))

  plot(result_seq$samples, pch = '.', main = "t-walk Samples (Sequential)")
}

# Example 2: named parameters and extra arguments
log_post_normal <- function(theta, y) {
  sum(dnorm(y, theta[["mu"]], exp(theta[["log_sigma"]]), log = TRUE))
}
y <- c(4.1, 5.3, 3.8, 4.9, 5.6)
fit <- twalk(log_post_normal, n_iter = 5000,
             x0 = c(mu = 4, log_sigma = 0), xp0 = c(mu = 5, log_sigma = 1),
             y = y, show_progress = FALSE)
summary(fit)


# Example 3: two chains in parallel from dispersed starting points
if (requireNamespace("mvtnorm", quietly = TRUE)) {
  set.seed(123)
  result_par <- twalk(log_posterior = log_post, n_iter = 2500,
                      x0 = rbind(c(-3, 3), c(3, -3)),
                      xp0 = rbind(c(-2, 2), c(2, -2)),
                      n_chains = 2, n_cores = 2)
  summary(result_par)
}
```

`twalk-methods` *Methods for objects of class 'twalk'*

Description

S3 methods for inspecting t-walk MCMC output objects. `print()` displays chain metadata and overall sampler-performance rates. `summary()` computes posterior summaries and convergence diagnostics (see `calculate_diagnostics()`) after discarding a burn-in fraction from each chain, and returns them as a data frame of class `summary.twalk`, whose print method also shows the per-kernel performance of the sampler.

Usage

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

## S3 method for class 'twalk'
summary(object, burnin_frac = 0.2, ...)

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

Arguments

x, object	An object of class twalk (or summary.twalk).
...	Unused.
burnin_frac	Fraction of each chain discarded as burn-in.

Value

print() returns x invisibly. summary() returns a data frame of class summary.twalk with one row per parameter.

Examples

```
log_post <- function(x) -0.5 * sum(x^2)
set.seed(1)
fit <- twalk(log_post, n_iter = 2000, x0 = c(a = -1, b = 1), xp0 = c(1, -1),
             show_progress = FALSE)
fit
s <- summary(fit, burnin_frac = 0.2)
s
s$Rhat
```

---

twalk_continue *Continue a t-walk run*

Description

Runs further iterations of a previous twalk() result, starting each chain from the final state of the sampler. The Markov chain is continued exactly: no burn-in or re-initialization takes place.

Usage

```
twalk_continue(
  object,
  log_posterior,
  n_iter,
```

```
  n_cores = NULL,
  show_progress = TRUE,
  append = TRUE,
  ...
)
```

Arguments

object	An object of class twalk.
log_posterior	The log-posterior function used to create object.
n_iter	Number of additional iterations per chain.
n_cores	Number of cores for multi-chain results (see twalk()).
show_progress	Logical; whether to display progress information.
append	Logical; if TRUE (default), the new iterations are appended to those in object. If FALSE, only the new iterations are returned.
...	Additional arguments passed to log_posterior.

Value

An object of class twalk.

Examples

```
log_post <- function(x) -0.5 * sum(x^2)
set.seed(1)
fit <- twalk(log_post, n_iter = 2000, x0 = c(-1, 1), xp0 = c(1, -1),
             show_progress = FALSE)
fit <- twalk_continue(fit, log_post, n_iter = 3000, show_progress = FALSE)
fit$n_iter
```

---

twalk_control *Control parameters for the t-walk*

Description

Collects the tuning constants of the t-walk. The defaults are the values recommended by Christen & Fox (2010, Section 2.5), obtained by minimizing integrated autocorrelation times over a wide range of targets; the authors' intention is that they are left unchanged. Changing them is mainly useful for sensitivity studies.

Usage

```
twalk_control(
  n1 = 4,
  aw = 1.5,
  at = 6,
  kernel_probs = c(traverse = 0.4918, walk = 0.4918, blow = 0.0082, hop = 0.0082),
  store_companion = TRUE,
  engine = c("cpp", "R")
)
```

Arguments

`n1`	Expected number of coordinates moved at each iteration. Each coordinate is updated with probability `min(n, n1) / n`, where `n` is the dimension (Section 2). The paper explored `n1` in `[2, 20]`.
`aw`	Walk-move parameter `a_w > 0` (Section 2.1). The paper explored `[0.3, 2]`.
`at`	Traverse-move parameter `a_t > 1` (Section 2.2). The paper explored `[2, 10]`.
`kernel_probs`	Probabilities of choosing the Traverse, Walk, Blow and Hop moves, in this order (or named). They are normalized to sum to one. The defaults correspond to the move ratios 60:60:1:1.
`store_companion`	Logical; whether to store the trajectory of the companion point `x'`. Setting it to `FALSE` halves the memory used by the output.
`engine`	Either `"cpp"` (default), the compiled engine, or `"R"`, a much slower pure-R reference implementation that produces identical chains and is used to validate the compiled engine.

Value

An object of class `twalk_control`.

References

Christen, J. A. and Fox, C. (2010). A general purpose sampling algorithm for continuous distributions (the t-walk). *Bayesian Analysis*, 5(2), 263–282. doi:10.1214/10BA603

Examples

```
twalk_control()

# A sensitivity study on the Walk-move parameter
twalk_control(aw = 0.5)
```

`twalk_step` *Perform a single t-walk transition*

Description

Applies one transition of the t-walk kernel to the pair `(x, xp)`. This is a building block for hybrid samplers: Christen & Fox (2010, Section 5) combine the t-walk kernel with Gibbs updates for blocks of parameters whose full conditionals are available.

Usage

```
twalk_step(
  log_posterior,
  x,
  xp,
  log_x = NULL,
  log_xp = NULL,
  control = twalk_control(),
  ...
)
```

Arguments

`log_posterior`	Log-density function, as in `twalk()`.
`x`, `xp`	The current two points.
`log_x`, `log_xp`	Optional log densities at `x` and `xp`.
`control`	A `twalk_control()` object.
`...`	Additional arguments passed to `log_posterior`.

Details

Since the t-walk targets $\pi(x)\pi(x')$, the log densities of both points are needed. Pass them through `log_x` and `log_xp` to avoid re-evaluating them; when the target changes between calls (as in a Gibbs sampler, where the conditional depends on the other blocks), leave them as `NULL` so they are recomputed.

Value

A list with the new points `x` and `xp`, their log densities `log_x` and `log_xp`, the `kernel` used, and the logical flags `accepted` and `moved`. If `log_posterior` raised an error at the proposal, the proposal is rejected and the message is returned in `error`.

Examples

```
# Gibbs sampler for (mu, tau) in a normal model, where the t-walk updates
# mu given tau and tau is drawn from its full conditional.
set.seed(1)
y <- rnorm(30, 2, 1.5)
log_cond_mu <- function(mu, tau) sum(dnorm(y, mu, 1 / sqrt(tau), log = TRUE))

mu <- 0; mu_p <- 1; tau <- 1
draws <- matrix(NA, 2000, 2, dimnames = list(NULL, c("mu", "tau")))
for (i in 1:2000) {
  step <- twalk_step(log_cond_mu, mu, mu_p, tau = tau)
  mu <- step$x; mu_p <- step$xp
  tau <- rgamma(1, 1 + length(y) / 2, 1 + sum((y - mu)^2) / 2)
  draws[i, ] <- c(mu, tau)
}
colMeans(draws[-(1:500), ])
```

---

visualize_results *Visualize MCMC results*

Description

Produces trace plots, marginal densities and joint plots depending on the dimension of the parameter space.

Usage

```
visualize_results(
  samples,
  true_values = NULL,
  title = "Results",
  burnin_frac = 0.2,
  true_covariance = NULL,
  show_acf = TRUE
)
```

Arguments

samples	Matrix of MCMC samples.
true_values	Optional vector of true parameter values.
title	Plot title.
burnin_frac	Burn-in fraction.
true_covariance	Optional true covariance matrix.
show_acf	Logical; whether to display autocorrelation plots.

Value

No return value; called for its side effect of drawing plots on the current graphics device.

Examples

```
log_post <- function(x) dnorm(x, log = TRUE)
res <- twalk(log_post, n_iter = 2000, x0 = -2, xp0 = 2)
visualize_results(
  res$samples,
  true_values = 0,
  title = "Standard normal"
)
```

Index

as.mcmc.list.twalk *(*coda-methods*)*, 3
as.mcmc.twalk *(*coda-methods*)*, 3
as_draws.twalk *(*posterior-methods*)*, 5
as_draws_array.twalk
*(*posterior-methods*)*, 5
as_draws_df.twalk *(*posterior-methods*)*, 5

calculate_diagnostics, 2
calculate_diagnostics(), *9*
coda-methods, 3

plot.twalk, 4
plot.twalk(), *8*
posterior-methods, 5
print.summary.twalk *(*twalk-methods*)*, 9
print.twalk *(*twalk-methods*)*, 9

summary.twalk *(*twalk-methods*)*, 9
summary.twalk(), *8*

twalk, 6
twalk(), *10, 11*, *13*
twalk-methods, 9
twalk_continue, 10
twalk_continue(), *8*
twalk_control, 11
twalk_control(), *7, 8*, *13*
twalk_step, 13
twalk_step(), *8*

visualize_results, 14