Package: gpumetropolis (via r-universe)

June 2, 2026

Title GPU-Portable Vendor-Agnostic Metropolis-Hastings Sampler

Version 0.2.0

Description A generic random-walk Metropolis-Hastings sampler for Markov chain Monte Carlo. The user declares a model by writing its log-likelihood and log-prior as one-sided formulas; the package compiles them to a stack-machine bytecode that a single portable kernel interprets, so the same model runs on the CPU and on the GPU from one source. The sampler advances many independent chains in one batched pass. A distributional equivalence harness based on the two-sample Kolmogorov-Smirnov test and the split R-hat statistic of Gelman and Rubin (1992) <doi:10.1214/ss/1177011136> checks the GPU output against the CPU output. Dispatching the kernel to a GPU helps when many chains are run or when the log-density is expensive to evaluate over a large data set, and does not help for small models run with few chains.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc >= 1.85.0, xz

Depends R (>= 4.2)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/pcbrom/gpumetropolis

BugReports https://github.com/pcbrom/gpumetropolis/issues

Config/pak/sysreqs xz-utils libclang-dev

Repository https://pcbrom.r-universe.dev

Date/Publication 2026-06-02 20:28:24 UTC

RemoteUrl https://github.com/pcbrom/gpumetropolis

RemoteRef HEAD

RemoteSha fe9a9fe0377cadb155f8adb1c943844eca43f901

Contents

`ess` *Effective sample size*

Description

Estimates the effective sample size of a set of chains: the number of independent draws that would carry the same information as the autocorrelated Markov chain Monte Carlo output. Each chain is reduced by Geyer's initial positive sequence estimator and the per-chain values are summed.

Usage

```
ess(x, warmup = NULL)
```

Arguments

`x`	A `gpumetropolis_fit` object, an `n_iter` by `n_chains` numeric matrix, or a numeric vector of draws.
`warmup`	Number of initial iterations discarded before the estimate. When `NULL`, half of the iterations are discarded for matrix and fit inputs, and none for vector inputs.

Value

A single numeric value, the summed effective sample size.

References

Geyer, C. J. (1992). Practical Markov chain Monte Carlo. Statistical Science 7(4), 473-483. doi:10.1214/ss/1177011137.

Examples

```
set.seed(1)
x <- rnorm(800, mean = 0, sd = 1)
fit <- metropolis_gaussian_mean(x, sigma = 1, n_iter = 2000)
ess(fit)
```

`gaussian_mean_posterior`

Closed-form posterior of the Gaussian mean

Description

Returns the analytic posterior of the mean `mu` for observations following `Normal(mu, sigma^2)` with known `sigma` and a flat prior on `mu`. The posterior is `Normal(mean(data), sigma^2 / length(data))`. It provides the ground truth used to check that `metropolis_gaussian_mean()` recovers known parameters.

Usage

```
gaussian_mean_posterior(data, sigma)
```

Arguments

`data`	Numeric vector of observations.
`sigma`	Positive finite scalar, the known standard deviation.

Value

A list with the posterior `mean` and standard deviation `sd`.

Examples

```
set.seed(1)
gaussian_mean_posterior(rnorm(500, mean = 3, sd = 2), sigma = 2)
```

gpu_metropolis *Run the GPU-portable batched Metropolis sampler*

Description

Advances many independent random-walk Metropolis chains over a model declared with gpum_model(). The log-density kernel runs on the chosen backend; the data is uploaded once and stays resident across the run.

Usage

```
gpu_metropolis(
  model,
  data = NULL,
  init = NULL,
  proposal_sd = 0.1,
  n_iter = 2000L,
  n_chains = 4L,
  warmup = NULL,
  adapt = TRUE,
  seed = 1L,
  backend = c("auto", "cpu", "cuda", "vulkan")
)
```

Arguments

model	A gpum_model object.
data	A named list or data frame with one entry per data column named in the model. Ignored for a model with no data term.
init	Optional numeric matrix of starting values, n_chains rows by one column per parameter. When supplied its row count sets the number of chains. When NULL, chains start from independent standard normal draws.
proposal_sd	Initial scale of the Gaussian random-walk proposal. Accepts a scalar (recycled to all parameters and chains), a length- n_params vector (broadcast across chains) or an n_chains by n_params matrix (one row per chain). When adapt = TRUE, this is only the warmup seed and adaptation refines it per chain.
n_iter	Iterations the sampler runs per chain, counting warmup. Default 2000.
n_chains	Number of chains. Used only when init is NULL. Default 4.
warmup	Warmup iterations to discard before returning, following the convention of Stan and nimble. Must lie in [0, n_iter). Default floor(n_iter / 2), so fit$draws is post-warmup and is suitable for direct plotting. Set warmup = 0 to keep every iteration, useful for inspecting the burn-in trajectory in a trace plot.
adapt	Whether to adapt the per-chain proposal scale during warmup. When TRUE (default) and warmup > 0, the warmup runs in batches; between batches, Welford

updates the per-chain running variance and Robbins-Monro updates the per-chain scalar toward the asymptotic optimum acceptance (0.234 in d >= 2, 0.44 in d = 1; Roberts-Gelman-Gilks 1997, Roberts-Rosenthal 2009). Adaptation stops cleanly at the end of warmup, so the sampling phase is stationary. Set FALSE to keep the trim-only warmup of 0.1.x.

seed — Integer seed. Each chain advances its own counter-based stream from a triple32 hash; the seed is itself hashed, so runs with consecutive integer seeds get independent streams.

backend — Compute backend: "cpu", "cuda" (NVIDIA-native), "vulkan" (vendor-agnostic, through wgpu), or "auto". "auto" selects "cuda" when its feature was compiled into the build, then "vulkan", and falls back to "cpu" with a one-shot per-session message stating that no GPU backend is available. Default "auto".

Value

An object of class gpum_fit: a list with draws (an n_iter - warmup by n_chains by n_params array of post-warmup samples), accept_rate, n_iter (kept count), n_iter_total (raw count), warmup and the rest of the run metadata.

See Also

gpum_model(), rhat(), ess()

Examples

```
set.seed(1)
y <- rnorm(2000, mean = 3, sd = 2)
m <- gpum_model(~ -((y - mu)^2) / 8, params = "mu", data = "y")
fit <- gpu_metropolis(m, data = list(y = y), proposal_sd = 0.05,
                      n_iter = 1000, n_chains = 4)
rhat(fit$draws[, , 1])
```

---

gpum_diagnose *One-call diagnostic: convergence stats, plots and verdict.*

Description

Produces a per-parameter table (mean, standard deviation, 2.5%, 50% and 97.5% quantiles, split R-hat, effective sample size and Monte Carlo standard error) and a convergence verdict from the asymptotic canonical thresholds (R-hat below 1.01 in every parameter and ESS at or above 400). When plot = TRUE, opens a multi-panel plot per parameter showing the trace, the pooled density, the running mean per chain and the pooled autocorrelation; when the fit carries adaptation bookkeeping, an extra panel shows the per-chain acceptance rate by warmup batch with the asymptotic optimum drawn as a horizontal reference.

Usage

```
gpum_diagnose(fit, plot = TRUE, return_data = FALSE)
```

Arguments

fit	A gpum_fit object from gpu_metropolis().
plot	Whether to open the diagnostic plot. Default TRUE.
return_data	Whether to return the structured stats invisibly. Default FALSE; the function then returns NULL invisibly.

Value

When return_data = TRUE, a list with summary (one row per parameter), verdict (the convergence diagnosis and a suggested next step), adaptation (passed through from fit$adaptation) and adaptation_hint (a character string when the last warmup batch closed below 80% of the asymptotic target acceptance, NULL otherwise). Otherwise NULL invisibly.

See Also

gpu_metropolis(), rhat(), ess()

---

gpum_model *Declare a model for the GPU-portable Metropolis sampler*

Description

Compiles a log-likelihood and an optional log-prior, declared as one-sided formulas, into the bytecode the sampler runs. The log-likelihood is a per-observation expression: the sampler sums it over the data. The formulas may use +, -, *, /, ^, unary -, and exp, log, sqrt; any other symbol or function is rejected with a clear error.

Usage

```
gpum_model(loglik, params, data = character(0), prior = NULL)
```

Arguments

loglik	One-sided formula, the per-observation log-likelihood, up to an additive constant. It may reference the parameter names and the data column names.
params	Character vector of parameter names.
data	Character vector of data column names. Empty for a model with no data term.
prior	One-sided formula, the joint log-prior over the parameters, up to an additive constant. It may reference only the parameter names. NULL is a flat prior.

Value

An object of class gpum_model.

See Also

gpu_metropolis()

Examples

```
# Gaussian mean with known sd = 2 and a flat prior on mu.
m <- gpum_model(
  loglik = ~ -((y - mu)^2) / 8,
  params = "mu",
  data = "y"
)
```

---

ks_equivalence *Distributional equivalence by the two-sample Kolmogorov-Smirnov test*

Description

Pools the post-warmup draws of two samplers and applies the two-sample Kolmogorov-Smirnov test. It is the equivalence gate for the package: a CPU and a GPU sampler are treated as equivalent when the test does not reject the hypothesis that their draws come from the same distribution.

Usage

```
ks_equivalence(x, y, warmup = NULL, alpha = 0.05, thin = TRUE)
```

Arguments

x, y	Each a gpumetropolis_fit object, an n_iter by n_chains numeric matrix, or a numeric vector of draws.
warmup	Number of initial iterations discarded from x and y before pooling. When NULL, half of the iterations are discarded for matrix and fit inputs, and none for vector inputs.
alpha	Significance level of the test. Default 0.05.
thin	Controls thinning before the test. TRUE (default) thins each pooled sample to its effective sample size. A positive integer keeps every thin-th draw. FALSE applies no thinning.

Details

The Kolmogorov-Smirnov test assumes independent draws, while Markov chain Monte Carlo output is autocorrelated; feeding it raw draws makes the test reject far too often. By default the pooled draws are therefore thinned down to the effective sample size returned by ess() before the test is applied.

Not rejecting is evidence consistent with equivalence, not a proof that the two distributions are identical. Report the statistic and the p-value, and read equivalent as "no detected difference at level alpha".

Value

A list with the test `statistic`, the `p_value`, the chosen `alpha`, a logical `equivalent` (`TRUE` when `p_value > alpha`) and the post-thinning pooled sample sizes `n_x` and `n_y`.

Examples

```
set.seed(1)
x <- rnorm(800, mean = 1, sd = 1)
a <- metropolis_gaussian_mean(x, sigma = 1, n_iter = 2000, seed = 1)
b <- metropolis_gaussian_mean(x, sigma = 1, n_iter = 2000, seed = 2)
ks_equivalence(a, b)
```

---

metropolis_gaussian_mean

Batched random-walk Metropolis sampler for a Gaussian mean

Description

Samples the posterior of the mean `mu` of observations assumed to follow `Normal(mu, sigma^2)` with known `sigma` and a flat prior on `mu`. Many independent chains are advanced together: at every iteration the candidate state of each chain is evaluated by a single batched call to the log-density kernel.

Usage

```
metropolis_gaussian_mean(
  data,
  sigma,
  n_iter = 2000L,
  n_chains = 4L,
  init = NULL,
  proposal_sd = NULL,
  seed = 1L
)
```

Arguments

`data`	Numeric vector of observations. Must be finite and non-empty.
`sigma`	Positive finite scalar, the known standard deviation of the observations.
`n_iter`	Number of iterations recorded per chain. Default 2000.
`n_chains`	Number of independent chains. Used only when `init` is `NULL`. Default 4.
`init`	Optional numeric vector of starting values, one per chain. When supplied, its length sets the number of chains. When `NULL`, chains start evenly spread over `mean(data) +/- 2 * sigma`, which overdisperses them so that `rhat()` is informative while staying bounded for many chains.

proposal_sd	Positive finite scalar, the standard deviation of the Gaussian random-walk proposal. When NULL, defaults to 2.4 * sigma / sqrt(length(data)), scaled to the width of the posterior.
seed	Single integer seed. Each chain runs an independent PCG64 stream derived from (seed, chain_index), so the result is reproducible and does not depend on chain scheduling.

Details

This is the CPU reference sampler. Its log-density kernel runs on the CPU. Later package versions dispatch the same kernel to a GPU and are validated for distributional equivalence against this function with ks_equivalence() and rhat(). The sequential dependence within a chain (x_{t+1} depends on x_t) is intrinsic to Markov chain Monte Carlo and is not parallelised; the parallel axes are the chains and the sum over the data.

Value

An object of class gpumetropolis_fit: a list with draws (an n_iter by n_chains numeric matrix), accept_rate (per-chain acceptance rate) and the run metadata.

See Also

rhat(), ess(), ks_equivalence(), gaussian_mean_posterior()

Examples

```
set.seed(1)
x <- rnorm(500, mean = 3, sd = 2)
fit <- metropolis_gaussian_mean(x, sigma = 2, n_iter = 1000)
rhat(fit)
```

---

rhat *Split R-hat convergence diagnostic*

Description

Computes the split potential scale reduction factor for a set of chains. Each chain is split in half and the halves are treated as separate chains, which makes the statistic sensitive to non-stationarity within a chain. A value near 1 is consistent with the chains having reached the same distribution; values above roughly 1.01 to 1.1 indicate that they have not.

Usage

```
rhat(x, warmup = NULL)
```

Arguments

x	A gpumetropolis_fit object or an n_iter by n_chains numeric matrix of draws.
warmup	Number of initial iterations discarded before the split. When NULL, half of the iterations are discarded.

Value

A single numeric value, the split R-hat statistic.

References

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. Statistical Science 7(4), 457-472. doi:10.1214/ss/1177011136.

Examples

```
set.seed(1)
x <- rnorm(500, mean = 0, sd = 1)
fit <- metropolis_gaussian_mean(x, sigma = 1, n_iter = 1000)
rhat(fit)
```

Index