Package: StanEstimators (via r-universe)

October 8, 2024

Title Estimate Parameters for Arbitrary R Functions using 'Stan'

Version 0.1.2.9000

Description Allows for the estimation of parameters for 'R' functions
using the various algorithms implemented in the 'Stan'
probabilistic programming language.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

NeedsCompilation yes

UseLTO true

SystemRequirements GNU make

Imports Rcpp, RcppParallel, posterior, methods, stats, loo

LinkingTo Rcpp, RcppEigen, BH, RcppParallel, rapidjsonr

Suggests testthat (>= 3.0.0), callr, future, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

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

RemoteUrl https://github.com/andrjohns/StanEstimators

RemoteRef HEAD

RemoteSha db7b96582b3f11c0ce66f9cd866d16fdc8a545b2

Contents

`StanEstimators-package` *The StanEstimators package.*

Description

ToDO

Author(s)

Maintainer: Andrew R. Johnson <`andrew.johnson@arjohnsonau.com`> (ORCID)

Other contributors:

- Jonah Gabry <`jsg2201@columbia.edu`> [contributor]
- Rok Češnovar <`rok.cesnovar@fri.uni-lj.si`> [contributor]
- Stan Development Team (CmdStan sources and headers) [copyright holder]
- Lawrence Livermore National Security (SUNDIALS sources and headers) [copyright holder]

`constrain_variables` *Constrain a vector of variables.*

Description

Constrain a vector of variables.

Usage

```
constrain_variables(stan_object, unconstrained_variables)

## S4 method for signature 'StanBase'
constrain_variables(stan_object, unconstrained_variables)
```

Arguments

`stan_object`	A `StanBase` object.
`unconstrained_variables`	Vector of unconstrained variables.

`grad_log_prob` *Calculate the log probability and its gradient of the model given a vector of unconstrained variables.*

Description

Calculate the log probability and its gradient of the model given a vector of unconstrained variables.

Usage

```
grad_log_prob(stan_object, unconstrained_variables, jacobian = TRUE)

## S4 method for signature 'StanBase'
grad_log_prob(stan_object, unconstrained_variables, jacobian = TRUE)
```

Arguments

`stan_object`	A `StanBase` object.
`unconstrained_variables`	Vector of unconstrained variables.
`jacobian`	Whether to include the Jacobian adjustment.

`hessian` *Calculate the log probability, its gradient, and its hessian matrix of the model given a vector of unconstrained variables.*

Description

Calculate the log probability, its gradient, and its hessian matrix of the model given a vector of unconstrained variables.

Usage

```
hessian(stan_object, unconstrained_variables, jacobian = TRUE)

## S4 method for signature 'StanBase'
hessian(stan_object, unconstrained_variables, jacobian = TRUE)
```

Arguments

`stan_object`	A `StanBase` object.
`unconstrained_variables`	Vector of unconstrained variables.
`jacobian`	Whether to include the Jacobian adjustment.

`log_prob` *Calculate the log probability of the model given a vector of unconstrained variables.*

Description

Calculate the log probability of the model given a vector of unconstrained variables.

Usage

```
log_prob(stan_object, unconstrained_variables, jacobian = TRUE)

## S4 method for signature 'StanBase'
log_prob(stan_object, unconstrained_variables, jacobian = TRUE)
```

Arguments

`stan_object`	A `StanBase` object.
`unconstrained_variables`	Vector of unconstrained variables.
`jacobian`	Whether to include the Jacobian adjustment.

`loo.StanBase` *Calculate approximate leave-one-out cross-validation (LOO-CV) for a model.*

Description

Calculate approximate leave-one-out cross-validation (LOO-CV) for a model.

Usage

```
## S3 method for class 'StanBase'
loo(x, pointwise_ll_fun, additional_args = list(), moment_match = FALSE, ...)
```

Arguments

`x`	A `StanBase` object.
`pointwise_ll_fun`	Function that calculates the pointwise log-likelihood given a vector of parameter values.
`additional_args`	List of additional arguments to be passed to `pointwise_ll_fun`.
`moment_match`	(logical) Whether to use a moment-matching correction for problematic observations.
`...`	Additional arguments to be passed to `loo::loo()`.

`StanBase-class` *StanBase base class*

Description

StanBase base class

`stan_diagnose`	*stan_diagnose*

Description

Check gradient estimation using Stan's 'Diagnose' method

Usage

```
stan_diagnose(
  fn,
  par_inits = NULL,
  n_pars = NULL,
  additional_args = list(),
  grad_fun = NULL,
  lower = -Inf,
  upper = Inf,
  eval_standalone = FALSE,
  globals = TRUE,
  packages = NULL,
  seed = NULL,
  refresh = NULL,
  quiet = FALSE,
  output_dir = NULL,
  output_basename = NULL,
  sig_figs = NULL
)
```

Arguments

`fn`	Function to estimate parameters for
`par_inits`	Initial values for parameters (must be specified if `n_pars` is NULL)
`n_pars`	Number of parameters to estimate (must be specified if `par_inits` is NULL)
`additional_args`	List of additional arguments to pass to the function
`grad_fun`	Function calculating gradients w.r.t. each parameter
`lower`	Lower bound constraint(s) for parameters
`upper`	Upper bound constraint(s) for parameters
`eval_standalone`	(logical) Whether to evaluate the function in a separate R session. Defaults to `FALSE`.
`globals`	(optional) a logical, a character vector, or a named list to control how globals are handled when evaluating functions in a separate R session. Ignored if `eval_standalone` = `FALSE`. For details, see section 'Globals used by future expressions' in the help for `future::future()`.

`packages`	(optional) a character vector specifying packages to be attached in the R environment evaluating the function. Ignored if `eval_standalone = FALSE`.
`seed`	Random seed
`refresh`	Number of iterations for printing
`quiet`	(logical) Whether to suppress Stan's output
`output_dir`	Directory to store outputs
`output_basename`	Basename to use for output files
`sig_figs`	Number of significant digits to use for printing

Value

`StanLaplace` object

`stan_laplace` *stan_laplace*

Description

Estimate parameters using Stan's laplace algorithm

Usage

```
stan_laplace(
  fn,
  par_inits = NULL,
  n_pars = NULL,
  additional_args = list(),
  grad_fun = NULL,
  lower = -Inf,
  upper = Inf,
  eval_standalone = FALSE,
  globals = TRUE,
  packages = NULL,
  seed = NULL,
  refresh = NULL,
  quiet = FALSE,
  output_dir = NULL,
  output_basename = NULL,
  sig_figs = NULL,
  mode = NULL,
  jacobian = NULL,
  draws = NULL,
  opt_args = NULL
)
```

Arguments

`fn`	Function to estimate parameters for
`par_inits`	Initial values for parameters (must be specified if `n_pars` is NULL)
`n_pars`	Number of parameters to estimate (must be specified if `par_inits` is NULL)
`additional_args`	List of additional arguments to pass to the function
`grad_fun`	Function calculating gradients w.r.t. each parameter
`lower`	Lower bound constraint(s) for parameters
`upper`	Upper bound constraint(s) for parameters
`eval_standalone`	(logical) Whether to evaluate the function in a separate R session. Defaults to `FALSE`.
`globals`	(optional) a logical, a character vector, or a named list to control how globals are handled when evaluating functions in a separate R session. Ignored if `eval_standalone` = `FALSE`. For details, see section 'Globals used by future expressions' in the help for `future::future()`.
`packages`	(optional) a character vector specifying packages to be attached in the R environment evaluating the function. Ignored if `eval_standalone` = `FALSE`.
`seed`	Random seed
`refresh`	Number of iterations for printing
`quiet`	(logical) Whether to suppress Stan's output
`output_dir`	Directory to store outputs
`output_basename`	Basename to use for output files
`sig_figs`	Number of significant digits to use for printing
`mode`	Mode for the laplace approximation, can either be a vector of values, a StanOptimize object, or NULL.
`jacobian`	(logical) Whether or not to use the Jacobian adjustment for constrained variables.
`draws`	(positive integer) Number of approximate posterior samples to draw and save.
`opt_args`	(named list) A named list of optional arguments to pass to `stan_optimize()` if `mode=NULL`.

Value

`StanLaplace` object

stan_optimize *stan_optimize*

Description

Estimate parameters using Stan's optimization algorithms

Usage

```
stan_optimize(
  fn,
  par_inits = NULL,
  n_pars = NULL,
  additional_args = list(),
  algorithm = "lbfgs",
  grad_fun = NULL,
  lower = -Inf,
  upper = Inf,
  eval_standalone = FALSE,
  globals = TRUE,
  packages = NULL,
  seed = NULL,
  refresh = NULL,
  quiet = FALSE,
  output_dir = NULL,
  output_basename = NULL,
  sig_figs = NULL,
  save_iterations = NULL,
  jacobian = NULL,
  init_alpha = NULL,
  iter = NULL,
  tol_obj = NULL,
  tol_rel_obj = NULL,
  tol_grad = NULL,
  tol_rel_grad = NULL,
  tol_param = NULL,
  history_size = NULL
)
```

Arguments

`fn`	Function to estimate parameters for
`par_inits`	Initial values for parameters (must be specified if `n_pars` is NULL)
`n_pars`	Number of parameters to estimate (must be specified if `par_inits` is NULL)
`additional_args`	List of additional arguments to pass to the function

`algorithm`	(string) The optimization algorithm. One of `"lbfgs"`, `"bfgs"`, or `"newton"`.
`grad_fun`	Function calculating gradients w.r.t. each parameter
`lower`	Lower bound constraint(s) for parameters
`upper`	Upper bound constraint(s) for parameters
`eval_standalone`	(logical) Whether to evaluate the function in a separate R session. Defaults to `FALSE`.
`globals`	(optional) a logical, a character vector, or a named list to control how globals are handled when evaluating functions in a separate R session. Ignored if `eval_standalone` = `FALSE`. For details, see section 'Globals used by future expressions' in the help for `future::future()`.
`packages`	(optional) a character vector specifying packages to be attached in the R environment evaluating the function. Ignored if `eval_standalone` = `FALSE`.
`seed`	Random seed
`refresh`	Number of iterations for printing
`quiet`	(logical) Whether to suppress Stan's output
`output_dir`	Directory to store outputs
`output_basename`	Basename to use for output files
`sig_figs`	Number of significant digits to use for printing
`save_iterations`	Save optimization iterations to output file
`jacobian`	(logical) Whether or not to use the Jacobian adjustment for constrained variables. For historical reasons, the default is `FALSE`, meaning optimization yields the (regularized) maximum likelihood estimate. Setting it to `TRUE` yields the maximum a posteriori estimate.
`init_alpha`	(positive real) The initial step size parameter.
`iter`	(positive integer) The maximum number of iterations.
`tol_obj`	(positive real) Convergence tolerance on changes in objective function value.
`tol_rel_obj`	(positive real) Convergence tolerance on relative changes in objective function value.
`tol_grad`	(positive real) Convergence tolerance on the norm of the gradient.
`tol_rel_grad`	(positive real) Convergence tolerance on the relative norm of the gradient.
`tol_param`	(positive real) Convergence tolerance on changes in parameter value.
`history_size`	(positive integer) The size of the history used when approximating the Hessian. Only available for L-BFGS.

Value

`StanOptimize` object

stan_pathfinder *stan_pathfinder*

Description

Estimate parameters using Stan's pathfinder algorithm

Usage

```
stan_pathfinder(
  fn,
  par_inits = NULL,
  n_pars = NULL,
  additional_args = list(),
  grad_fun = NULL,
  lower = -Inf,
  upper = Inf,
  eval_standalone = FALSE,
  globals = TRUE,
  packages = NULL,
  seed = NULL,
  refresh = NULL,
  quiet = FALSE,
  output_dir = NULL,
  output_basename = NULL,
  sig_figs = NULL,
  init_alpha = NULL,
  tol_obj = NULL,
  tol_rel_obj = NULL,
  tol_grad = NULL,
  tol_rel_grad = NULL,
  tol_param = NULL,
  history_size = NULL,
  num_psis_draws = NULL,
  num_paths = NULL,
  save_single_paths = NULL,
  max_lbfgs_iters = NULL,
  num_draws = NULL,
  num_elbo_draws = NULL
)
```

Arguments

`fn`	Function to estimate parameters for
`par_inits`	Initial values for parameters (must be specified if `n_pars` is NULL)
`n_pars`	Number of parameters to estimate (must be specified if `par_inits` is NULL)

`additional_args`	List of additional arguments to pass to the function
`grad_fun`	Function calculating gradients w.r.t. each parameter
`lower`	Lower bound constraint(s) for parameters
`upper`	Upper bound constraint(s) for parameters
`eval_standalone`	(logical) Whether to evaluate the function in a separate R session. Defaults to `FALSE`.
`globals`	(optional) a logical, a character vector, or a named list to control how globals are handled when evaluating functions in a separate R session. Ignored if `eval_standalone` = `FALSE`. For details, see section 'Globals used by future expressions' in the help for `future::future()`.
`packages`	(optional) a character vector specifying packages to be attached in the R environment evaluating the function. Ignored if `eval_standalone` = `FALSE`.
`seed`	Random seed
`refresh`	Number of iterations for printing
`quiet`	(logical) Whether to suppress Stan's output
`output_dir`	Directory to store outputs
`output_basename`	Basename to use for output files
`sig_figs`	Number of significant digits to use for printing
`init_alpha`	(positive real) The initial step size parameter.
`tol_obj`	(positive real) Convergence tolerance on changes in objective function value.
`tol_rel_obj`	(positive real) Convergence tolerance on relative changes in objective function value.
`tol_grad`	(positive real) Convergence tolerance on the norm of the gradient.
`tol_rel_grad`	(positive real) Convergence tolerance on the relative norm of the gradient.
`tol_param`	(positive real) Convergence tolerance on changes in parameter value.
`history_size`	(positive integer) The size of the history used when approximating the Hessian.
`num_psis_draws`	(positive integer) Number PSIS draws to return.
`num_paths`	(positive integer) Number of single pathfinders to run.
`save_single_paths`	(logical) Whether to save the results of single pathfinder runs in multi-pathfinder.
`max_lbfgs_iters`	(positive integer) The maximum number of iterations for LBFGS.
`num_draws`	(positive integer) Number of draws to return after performing pareto smooted importance sampling (PSIS).
`num_elbo_draws`	(positive integer) Number of draws to make when calculating the ELBO of the approximation at each iteration of LBFGS.

Value

`StanPathfinder` object

stan_sample *stan_sample*

Description

Estimate parameters using Stan's sampling algorithms

Usage

```
stan_sample(
  fn,
  par_inits = NULL,
  n_pars = NULL,
  additional_args = list(),
  algorithm = "hmc",
  engine = "nuts",
  grad_fun = NULL,
  lower = -Inf,
  upper = Inf,
  eval_standalone = (parallel_chains > 1),
  globals = TRUE,
  packages = NULL,
  seed = NULL,
  refresh = NULL,
  quiet = FALSE,
  output_dir = NULL,
  output_basename = NULL,
  sig_figs = NULL,
  num_chains = 4,
  parallel_chains = 1,
  num_samples = 1000,
  num_warmup = 1000,
  save_warmup = NULL,
  thin = NULL,
  adapt_engaged = NULL,
  adapt_gamma = NULL,
  adapt_delta = NULL,
  adapt_kappa = NULL,
  adapt_t0 = NULL,
  adapt_init_buffer = NULL,
  adapt_term_buffer = NULL,
  adapt_window = NULL,
  int_time = NULL,
  max_treedepth = NULL,
  metric = NULL,
  metric_file = NULL,
  stepsize = NULL,
  stepsize_jitter = NULL
)
```

Arguments

`fn`	Function to estimate parameters for
`par_inits`	Initial values for parameters (must be specified if `n_pars` is NULL)
`n_pars`	Number of parameters to estimate (must be specified if `par_inits` is NULL)
`additional_args`	List of additional arguments to pass to the function
`algorithm`	(string) The sampling algorithm. One of `"hmc"` or `"fixed_param"`.
`engine`	(string) The `HMC` engine to use, one of `"nuts"` or `"static"`
`grad_fun`	Function calculating gradients w.r.t. each parameter
`lower`	Lower bound constraint(s) for parameters
`upper`	Upper bound constraint(s) for parameters
`eval_standalone`	(logical) Whether to evaluate the function in a separate R session. Defaults to `(parallel_chains > 1)`. Must be `TRUE` if `parallel_chains > 1`.
`globals`	(optional) a logical, a character vector, or a named list to control how globals are handled when evaluating functions in a separate R session. Ignored if `eval_standalone = FALSE`. For details, see section 'Globals used by future expressions' in the help for `future::future()`.
`packages`	(optional) a character vector specifying packages to be attached in the R environment evaluating the function. Ignored if `eval_standalone = FALSE`.
`seed`	Random seed
`refresh`	Number of iterations for printing
`quiet`	(logical) Whether to suppress Stan's output
`output_dir`	Directory to store outputs
`output_basename`	Basename to use for output files
`sig_figs`	Number of significant digits to use for printing
`num_chains`	(positive integer) The number of Markov chains to run. The default is 4.
`parallel_chains`	(positive integer) The number of chains to run in parallel, the default is 1.
`num_samples`	(positive integer) The number of post-warmup iterations to run per chain.
`num_warmup`	(positive integer) The number of warmup iterations to run per chain.
`save_warmup`	(logical) Should warmup iterations be saved? The default is `FALSE`.
`thin`	(positive integer) The period between saved samples. This should typically be left at its default (no thinning) unless memory is a problem.
`adapt_engaged`	(logical) Do warmup adaptation? The default is `TRUE`.
`adapt_gamma`	(positive real) Adaptation regularization scale.

`adapt_delta`	(real in `(0,1)`) The adaptation target acceptance statistic.
`adapt_kappa`	(positive real) Adaptation relaxation exponent.
`adapt_t0`	(positive real) Adaptation iteration offset.
`adapt_init_buffer`	(nonnegative integer) Width of initial fast timestep adaptation interval during warmup.
`adapt_term_buffer`	(nonnegative integer) Width of final fast timestep adaptation interval during warmup.
`adapt_window`	(nonnegative integer) Initial width of slow timestep/metric adaptation interval.
`int_time`	(positive real) Total integration time
`max_treedepth`	(positive integer) The maximum allowed tree depth for the NUTS engine.
`metric`	(string) One of `"diag_e"`, `"dense_e"`, or `"unit_e"`, specifying the geometry of the base manifold.
`metric_file`	(character vector) The paths to JSON or Rdump files (one per chain) compatible with CmdStan that contain precomputed inverse metrics.
`stepsize`	(positive real) The *initial* step size for the discrete approximation to continuous Hamiltonian dynamics.
`stepsize_jitter`	(real in `(0,1)`) Allows step size to be “jittered” randomly during sampling to avoid any poor interactions with a fixed step size and regions of high curvature.

Value

`StanMCMC` object

`stan_variational` *stan_variational*

Description

Estimate parameters using Stan’s variational inference algorithms

Usage

```
stan_variational(
  fn,
  par_inits = NULL,
  n_pars = NULL,
  additional_args = list(),
  algorithm = "meanfield",
  grad_fun = NULL,
  lower = -Inf,
  upper = Inf,
  eval_standalone = FALSE,
  globals = TRUE,
  packages = NULL,
  seed = NULL,
  refresh = NULL,
  quiet = FALSE,
  output_dir = NULL,
  output_basename = NULL,
  sig_figs = NULL,
  iter = NULL,
  grad_samples = NULL,
  elbo_samples = NULL,
  eta = NULL,
  adapt_engaged = NULL,
  adapt_iter = NULL,
  tol_rel_obj = NULL,
  eval_elbo = NULL,
  output_samples = NULL
)
```

Arguments

`fn`	Function to estimate parameters for
`par_inits`	Initial values for parameters (must be specified if `n_pars` is NULL)
`n_pars`	Number of parameters to estimate (must be specified if `par_inits` is NULL)
`additional_args`	List of additional arguments to pass to the function
`algorithm`	(string) The variational inference algorithm. One of `"meanfield"` or `"fullrank"`.
`grad_fun`	Function calculating gradients w.r.t. each parameter
`lower`	Lower bound constraint(s) for parameters
`upper`	Upper bound constraint(s) for parameters
`eval_standalone`	(logical) Whether to evaluate the function in a separate R session. Defaults to `FALSE`.
`globals`	(optional) a logical, a character vector, or a named list to control how globals are handled when evaluating functions in a separate R session. Ignored if `eval_standalone` = `FALSE`. For details, see section 'Globals used by future expressions' in the help for `future::future()`.
`packages`	(optional) a character vector specifying packages to be attached in the R environment evaluating the function. Ignored if `eval_standalone` = `FALSE`.
`seed`	Random seed
`refresh`	Number of iterations for printing
`quiet`	(logical) Whether to suppress Stan's output
`output_dir`	Directory to store outputs

`output_basename`	Basename to use for output files
`sig_figs`	Number of significant digits to use for printing
`iter`	(positive integer) The *maximum* number of iterations.
`grad_samples`	(positive integer) The number of samples for Monte Carlo estimate of gradients.
`elbo_samples`	(positive integer) The number of samples for Monte Carlo estimate of ELBO (objective function).
`eta`	(positive real) The step size weighting parameter for adaptive step size sequence.
`adapt_engaged`	(logical) Do warmup adaptation?
`adapt_iter`	(positive integer) The *maximum* number of adaptation iterations.
`tol_rel_obj`	(positive real) Convergence tolerance on the relative norm of the objective.
`eval_elbo`	(positive integer) Evaluate ELBO every Nth iteration.
`output_samples`	(positive integer) Number of approximate posterior samples to draw and save.

Value

`StanVariational` object

`stan_versions` *stan_versions*

Description

stan_versions

Usage

```
stan_versions()
```

Value

A named list with the Stan and Stan Math library versions

summary-StanLaplace *Summary method for objects of class* `StanLaplace`.

Description

Summary method for objects of class `StanLaplace`.

Usage

```
## S4 method for signature 'StanLaplace'
summary(object, ...)
```

Arguments

`object`	A `StanLaplace` object.
`...`	Additional arguments, currently unused.

summary-StanMCMC *Summary method for objects of class* `StanMCMC`.

Description

Summary method for objects of class `StanMCMC`.

Usage

```
## S4 method for signature 'StanMCMC'
summary(object, ...)
```

Arguments

`object`	A `StanMCMC` object.
`...`	Additional arguments, currently unused.

`summary-StanOptimize` *Summary method for objects of class* `StanOptimize`.

Description

Summary method for objects of class `StanOptimize`.

Usage

```
## S4 method for signature 'StanOptimize'
summary(object, ...)
```

Arguments

`object`	A `StanOptimize` object.
`...`	Additional arguments, currently unused.

`summary-StanPathfinder` *Summary method for objects of class* `StanPathfinder`.

Description

Summary method for objects of class `StanPathfinder`.

Usage

```
## S4 method for signature 'StanPathfinder'
summary(object, ...)
```

Arguments

`object`	A `StanPathfinder` object.
`...`	Additional arguments, currently unused.

summary-StanVariational *Summary method for objects of class* StanVariational.

Description

Summary method for objects of class StanVariational.

Usage

```
## S4 method for signature 'StanVariational'
summary(object, ...)
```

Arguments

object	A StanVariational object.
...	Additional arguments, currently unused.

unconstrain_draws *Unconstrain all parameter draws.*

Description

Unconstrain all parameter draws.

Usage

```
unconstrain_draws(stan_object, draws = NULL)

## S4 method for signature 'StanBase'
unconstrain_draws(stan_object, draws = NULL)

## S4 method for signature 'StanOptimize'
unconstrain_draws(stan_object, draws = NULL)
```

Arguments

stan_object	A StanBase object.
draws	(optional) A posterior::draws_* object to be unconstrained (instead of the draws in the StanBase object).

`unconstrain_variables` *Unconstrain a vector of variables.*

Description

Unconstrain a vector of variables.

Usage

```
unconstrain_variables(stan_object, variables)

## S4 method for signature 'StanBase'
unconstrain_variables(stan_object, variables)
```

Arguments

`stan_object`	A `StanBase` object.
`variables`	Vector of variables to be unconstrained.

Index