Package 'adnuts'

July 22, 2025

Title No-U-Turn MCMC Sampling for 'ADMB' Models

Version 1.1.2

Description Bayesian inference using the no-U-turn (NUTS) algorithm by Hoffman and Gelman (2014) <https://www.jmlr.org/papers/v15/hoffman14a.html>. Designed for 'AD Model Builder' ('ADMB') models, or when R functions for log-density and log-density gradient are available, such as 'Template Model Builder' models and other special cases. Functionality is similar to 'Stan', and the 'rstan' and 'shinystan' packages are used for diagnostics and inference.

Depends R (>= 3.6.0), snowfall (>= 1.84.6.1)

URL https://github.com/Cole-Monnahan-NOAA/adnuts

BugReports https://github.com/Cole-Monnahan-NOAA/adnuts/issues

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

ByteCompile true

Suggests shinystan (>= 2.5.0), matrixcalc (>= 1.0.3), stats, knitr, TMB, rmarkdown, withr, testthat (>= 2.1.0)

Imports ellipse, rstan, R2admb, ggplot2, rlang

VignetteBuilder knitr

NeedsCompilation no

Author Cole Monnahan [aut, cre] (ORCID: <https://orcid.org/0000-0003-0871-6700>)

Maintainer Cole Monnahan <monnahc@uw.edu>

Repository CRAN

Date/Publication 2021-03-02 19:50:09 UTC

Contents

`.check_ADMB_version` *Check that the model is compiled with the right version of ADMB which is 12.0 or later*

Description

Check that the model is compiled with the right version of ADMB which is 12.0 or later

Usage

```
.check_ADMB_version(model, path = getwd(), min.version = 12, warn = TRUE)
```

Arguments

model	Model name without file extension
path	Path to model folder, defaults to working directory. NULL value specifies working directory (default).
min.version	Minimum valid version (numeric). Defaults to 12.0.
warn	Boolean whether to throw warnings or not

Details

Some functionality of packages **adnuts** is imbedded in the ADMB source code so that when a model is compiled it is contained in the model executable. If this code does not exist adnuts will fail. The solution is to update ADMB and recompile the model.

Value

Nothing, errors out if either model could not be run or the version is incompatible. If compatible nothing happens.

.check_console_printing

Check if the session is interactive or Rstudio which has implications for parallel output

---

Description

Check if the session is interactive or Rstudio which has implications for parallel output

Usage

```
.check_console_printing(parallel)
```

Arguments

parallel	Boolean whether chain is executed in parallel mode or not.

Details

When using RStudio and RGui, the parallel output does not show on the console. As a workaround it is captured in each cluster into a file and then read in and printed.

Value

Boolean whether output should be printed to console progressively, or saved to file and printed at the end.

.check_model_path *Check that the file can be found*

Description

Check that the file can be found

Usage

```
.check_model_path(model, path)
```

Arguments

model	Model name without file extension
path	Path to model folder, defaults to working

.getADMBHessian *Read in admodel.hes file*

Description

Read in admodel.hes file

Usage

```
.getADMBHessian(path)
```

Arguments

path	Path to folder containing the admodel.hes file

Value

The Hessian matrix

`.sample_admb` *Hidden wrapper function for sampling from ADMB models*

Description

Hidden wrapper function for sampling from ADMB models

Usage

```
.sample_admb(
  model,
  path = getwd(),
  iter = 2000,
  init = NULL,
  chains = 3,
  warmup = NULL,
  seeds = NULL,
  thin = 1,
  mceval = FALSE,
  duration = NULL,
  cores = NULL,
  control = NULL,
  verbose = TRUE,
  algorithm = "NUTS",
  skip_optimization = TRUE,
  skip_monitor = FALSE,
  skip_unbounded = TRUE,
  admb_args = NULL
)
```

Arguments

`model`	Name of model (i.e., 'model' for model.tpl). For non-Windows systems this will automatically be converted to './model' internally. For Windows, long file names are sometimes shortened from e.g., 'long_model_filename' to 'LONG_~1'. This should work, but will throw warnings. Please shorten the model name. See https://en.wikipedia.org/wiki/8.3_filename.
`path`	Path to model executable. Defaults to working directory. Often best to have model files in a separate subdirectory, particularly for parallel.
`iter`	The number of samples to draw.
`init`	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., `obj$par`) for all chains.
`chains`	The number of chains to run.

warmup	The number of warmup iterations.
seeds	A vector of seeds, one for each chain.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
mceval	Whether to run the model with -mceval on samples from merged chains.
duration	The number of minutes after which the model will quit running.
cores	The number of cores to use for parallel execution. Default is number available in the system minus 1. If cores=1, serial execution occurs (even if chains>1), otherwise parallel execution via package snowfall is used. For slow analyses it is recommended to set chains<=cores so each core needs to run only a single chain.
control	A list to control the sampler. See details for further use.
verbose	Flag whether to show console output (default) or suppress it completely except for warnings and errors. Works for serial or parallel execution.
algorithm	The algorithm to use, one of "NUTS" or "RWM"
skip_optimization	Whether to run the optimizer before running MCMC. This is rarely need as it is better to run it once before to get the covariance matrix, or the estimates are not needed with adaptive NUTS.
skip_monitor	Whether to skip calculating diagnostics (effective sample size, Rhat) via the rstan::monitor function. This can be slow for models with high dimension or many iterations. The result is used in plots and summaries so it is recommended to turn on. If model run with skip_monitor=FALSE you can recreate it post-hoc by setting fit$monitor=rstan::monitor(fit$samples,fit$warmup, print=FALSE).
skip_unbounded	Whether to skip returning the unbounded version of the posterior samples in addition to the bounded ones. It may be advisable to set to FALSE for very large models to save space.
admb_args	A character string which gets passed to the command line, allowing finer control

---

.update_model *Convert model name depending on system*

Description

Convert model name depending on system

Usage

```
.update_model(model)
```

Arguments

model	Model name without file extension

Value

Updated model name to use with system call

`adfit` *Constructor for the "adfit" (A-D fit) class*

Description

Constructor for the "adfit" (A-D fit) class

Usage

```
adfit(x)
```

Arguments

`x`	Fitted object from `sample_admb`

Value

An object of class "adfit"

`adnuts` *adnuts: No-U-turn sampling for AD Model Builder (ADMB)*

Description

Draw Bayesian posterior samples from an ADMB model using the no-U-turn MCMC sampler. Adaptation schemes are used so specifying tuning parameters is not necessary, and parallel execution reduces overall run time.

Details

The software package Stan pioneered the use of no-U-turn (NUTS) sampling for Bayesian models (Hoffman and Gelman 2014, Carpenter et al. 2017). This algorithm provides fast, efficient sampling across a wide range of models, including hierarchical ones, and thus can be used as a generic modeling tool (Monnahan et al. 2017). The functionality provided by **adnuts** is based loosely off Stan and R package **rstan**

The **adnuts** R package provides an R workflow for NUTS sampling for ADMB models (Fournier et al. 2011), including adaptation of step size and metric (mass matrix), parallel execution, and links to diagnostic and inference tools provided by **rstan** and **shinystan**. The ADMB implementation of NUTS code is bundled into the ADMB source itself (as of version 12.0). Thus, when a user builds an ADMB model the NUTS code is incorporated into the model executable. Thus, **adnuts** simply provides a convenient set of wrappers to more easily execute, diagnose, and make inference on a model. More details can be found in the package vignette.

Note that previous versions of **adnuts** included functionality for TMB models, but this has been replaced by **tmbstan** (Kristensen et al. 2016, Monnahan and Kristensen 2018).

References

Carpenter, B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Riddell, A., Guo, J.Q., Li, P., Riddell, A., 2017. Stan: A Probabilistic Programming Language. J Stat Softw. 76:1-29.

Fournier, D.A., Skaug, H.J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M.N., Nielsen, A., Sibert, J., 2012. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optim Method Softw. 27:233-249.

Hoffman, M.D., Gelman, A., 2014. The no-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J Mach Learn Res. 15:1593-1623.

Kristensen, K., Nielsen, A., Berg, C.W., Skaug, H., Bell, B.M., 2016. TMB: Automatic differentiation and Laplace approximation. J Stat Softw. 70:21.

Kristensen, K., 2017. TMB: General random effect model builder tool inspired by ADMB. R package version 1.7.11.

Monnahan, C.C., Thorson, J.T., Branch, T.A., 2017. Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo. Methods in Ecology and Evolution. 8:339-348.

Monnahan C.C., Kristensen K. (2018). No-U-turn sampling for fast Bayesian inference in ADMB and TMB: Introducing the adnuts and tmbstan R packages PLoS ONE 13(5): e0197954. https://doi.org/10.1371/journal.pone.

Stan Development Team, 2016. Stan modeling language users guide and reference manual, version 2.11.0.

Stan Development Team, 2016. RStan: The R interface to Stan. R package version 2.14.1. http://mc-stan.org.

`as.data.frame.adfit` *Convert object of class adfit to data.frame. Calls* `extract_samples`

Description

Convert object of class adfit to data.frame. Calls `extract_samples`

Usage

```
## S3 method for class 'adfit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	Fitted object from `sample_rwm`
`row.names`	Ignored
`optional`	Ignored
`...`	Ignored

Details

This calls the default settings of `extract_samples`, no warmup samples and no column for the log-posterior (lp__). Use this function directly for finer control.

Value

A data frame with parameters as columns and samples as rows.

---

check_identifiable *Check identifiability from model Hessian*

Description

Check identifiability from model Hessian

Usage

```
check_identifiable(model, path = getwd())
```

Arguments

`model`	Model name without file extension
`path`	Path to model folder, defaults to working directory

Details

Read in the admodel.hes file and check the eigenvalues to determine which parameters are not identifiable and thus cause the Hessian to be non-invertible. Use this to identify which parameters are problematic. This function was converted from a version in the `FishStatsUtils` package.

Value

Prints output of bad parameters and invisibly returns it.

`extract_sampler_params` *Extract sampler parameters from a fit.*

Description

Extract information about NUTS trajectories, such as acceptance ratio and treedepth, from a fitted object.

Usage

```
extract_sampler_params(fit, inc_warmup = FALSE)
```

Arguments

`fit`	A list returned by `sample_admb`.
`inc_warmup`	Whether to extract the warmup samples or not (default). Warmup samples should never be used for inference, but may be useful for diagnostics.

Details

Each trajectory (iteration) in NUTS has associated information about the trajectory: stepsize, acceptance ratio, treedepth, and number of leapfrog steps. This function extracts these into a data.frame, which may be useful for diagnosing issues in certain cases. In general, the user should not need to examine them, or preferably should via `plot_sampler_params` or `launch_shinyadmb`.

Value

An invisible data.frame containing samples (rows) of each parameter (columns). If multiple chains exist they will be rbinded together.

See Also

`launch_shinyadmb`.

Examples

```
fit <- readRDS(system.file('examples', 'fit.RDS', package='adnuts'))
sp <- extract_sampler_params(fit, inc_warmup=TRUE)
str(sp)
```

`extract_samples` *Extract posterior samples from a model fit.*

Description

A helper function to extract posterior samples across multiple chains into a single data.frame.

Usage

```
extract_samples(
  fit,
  inc_warmup = FALSE,
  inc_lp = FALSE,
  as.list = FALSE,
  unbounded = FALSE
)
```

Arguments

`fit`	A list returned by `sample_admb`.
`inc_warmup`	Whether to extract the warmup samples or not (default). Warmup samples should never be used for inference, but may be useful for diagnostics.
`inc_lp`	Whether to include a column for the log posterior density (last column). For diagnostics it can be useful.
`as.list`	Whether to return the samples as a list (one element per chain). This could then be converted to a CODA mcmc object.
`unbounded`	Boolean flag whether to return samples in unbounded (untransformed) space. Will only be differences when init_bounded types are used in the ADMB template. This can be useful for model debugging.

Details

This function is loosely based on the **rstan** function `extract`. Merging samples across chains should only be used for inference after appropriate diagnostic checks. Do not calculate diagnostics like Rhat or effective sample size after using this function, instead, use `monitor`. Likewise, warmup samples are not valid and should never be used for inference, but may be useful in some cases for diagnosing issues.

Value

If as.list is FALSE, an invisible data.frame containing samples (rows) of each parameter (columns). If multiple chains exist they will be rbinded together, maintaining order within each chain. If as.list is TRUE, samples are returned as a list of matrices.

Examples

```
## A previously run fitted ADMB model
fit <- readRDS(system.file('examples', 'fit.RDS', package='adnuts'))
post <- extract_samples(fit)
tail(apply(post, 2, median))
```

`is.adfit` *Check object of class adfit*

Description

Check object of class adfit

Usage

```
is.adfit(x)
```

Arguments

`x`	Returned list from `sample_admb`

`launch_shinyadmb` *Launch shinystan for an ADMB fit.*

Description

Launch shinystan for an ADMB fit.

Usage

```
launch_shinyadmb(fit)
```

Arguments

`fit`	A named list returned by `sample_admb`.

See Also

`launch_shinytmb`

launch_shinytmb *Launch shinystan for a TMB fit.*

Description

Launch shinystan for a TMB fit.

Usage

```
launch_shinytmb(fit)
```

Arguments

fit	A named list returned by sample_tmb.

See Also

launch_shinyadmb

pairs_admb *Plot pairwise parameter posteriors and optionally the MLE points and confidence ellipses.*

Description

Plot pairwise parameter posteriors and optionally the MLE points and confidence ellipses.

Usage

```
pairs_admb(
  fit,
  order = NULL,
  diag = c("trace", "acf", "hist"),
  acf.ylim = c(-1, 1),
  ymult = NULL,
  axis.col = gray(0.5),
  pars = NULL,
  label.cex = 0.8,
  limits = NULL,
  add.mle = TRUE,
  add.monitor = TRUE,
  unbounded = FALSE,
  ...
)
```

Arguments

fit	A list as returned by sample_admb.
order	The order to consider the parameters. Options are NULL (default) to use the order declared in the model, or 'slow' and 'fast' which are based on the effective sample sizes ordered by slowest or fastest mixing respectively. See example for usage.
diag	What type of plot to include on the diagonal, options are 'acf' which plots the autocorrelation function acf, 'hist' shows marginal posterior histograms, and 'trace' the trace plot.
acf.ylim	If using the acf function on the diagonal, specify the y limit. The default is c(-1,1).
ymult	A vector of length ncol(posterior) specifying how much room to give when using the hist option for the diagonal. For use if the label is blocking part of the plot. The default is 1.3 for all parameters.
axis.col	Color of axes
pars	A vector of parameter names or integers representing which parameters to subset. Useful if the model has a larger number of parameters and you just want to show a few key ones.
label.cex	Control size of outer and diagonal labels (default 1)
limits	A list containing the ranges for each parameter to use in plotting.
add.mle	Boolean whether to add 95% confidence ellipses
add.monitor	Boolean whether to print effective sample
unbounded	Whether to use the bounded or unbounded version of the parameters. size (ESS) and Rhat values on the diagonal.
...	Arguments to be passed to plot call in lower diagonal panels

Details

This function is modified from the base pairs code to work specifically with fits from the 'adnuts' package using either the NUTS or RWM MCMC algorithms. If an invertible Hessian was found (in fit$mle) then estimated covariances are available to compare and added automatically (red ellipses). Likewise, a "monitor" object from rstan::monitor is attached as fit$monitor and provides effective sample sizes (ESS) and Rhat values. The ESS are used to potentially order the parameters via argument order, but also printed on the diagonal.

Value

Produces a plot, and returns nothing.

Author(s)

Cole Monnahan

Examples

```
fit <- readRDS(system.file('examples', 'fit.RDS', package='adnuts'))
pairs_admb(fit)
pairs_admb(fit, pars=1:2)
pairs_admb(fit, pars=c('b', 'a'))
pairs_admb(fit, pars=1:2, order='slow')
pairs_admb(fit, pars=1:2, order='fast')
```

---

plot.adfit *Plot object of class adfit*

Description

Plot object of class adfit

Usage

```
## S3 method for class 'adfit'
plot(x, y, ...)
```

Arguments

x	Fitted object from sample_admb
y	Ignored
...	Ignored

Value

Plot created

---

plot_marginals *Plot marginal distributions for a fitted model*

Description

Plot marginal distributions for a fitted model

Usage

```
plot_marginals(
  fit,
  pars = NULL,
  mfrow = NULL,
  add.mle = TRUE,
  add.monitor = TRUE,
  breaks = 30
)
```

Arguments

fit	A fitted object returned by sample_admb.
pars	A numeric or character vector of parameters which to plot, for plotting a subset of the total (defaults to all)
mfrow	A custom grid size (vector of two) to be called as par(mfrow), overriding the defaults.
add.mle	Whether to add marginal normal distributions determined from the inverse Hessian file
add.monitor	Whether to add ESS and Rhat information
breaks	The number of breaks to use in hist(), defaulting to 30

Details

This function plots grid cells of all parameters in a model, comparing the marginal posterior histogram vs the asymptotic normal (red lines) from the inverse Hessian. Its intended use is to quickly gauge differences between frequentist and Bayesian inference on the same model.

If fit$monitor exists the effective sample size (ESS) and R-hat estimates are printed in the top right corner. See https://mc-stan.org/rstan/reference/Rhat.html for more information. Generally Rhat>1.05 or ESS<100 (per chain) suggest inference may be unreliable.

This function is customized to work with multipage PDFs, specifically: pdf('marginals.pdf', onefile=TRUE, width=7,height=5) produces a nice readable file.

Examples

```
fit <- readRDS(system.file('examples', 'fit.RDS', package='adnuts'))
plot_marginals(fit, pars=1:2)
```

---

plot_sampler_params *Plot adaptation metrics for a fitted model.*

Description

Plot adaptation metrics for a fitted model.

Usage

```
plot_sampler_params(fit, plot = TRUE)
```

Arguments

fit	A fitted object returned by sample_admb.
plot	Whether to plot the results

Details

This utility function quickly plots the adaptation output of NUTS chains.

Value

Prints and invisibly returns a ggplot object

Examples

```
fit <- readRDS(system.file('examples', 'fit.RDS', package='adnuts'))
plot_sampler_params(fit)
```

---

plot_uncertainties *Plot MLE vs MCMC marginal standard deviations for each parameter*

Description

Plot MLE vs MCMC marginal standard deviations for each parameter

Usage

```
plot_uncertainties(fit, log = TRUE, plot = TRUE)
```

Arguments

fit	A fitted object returned by sample_admb
log	Whether to plot the logarithm or not.
plot	Whether to plot it or not.

Details

It can be helpful to compare uncertainty estimates between the two paradigms. This plots the marginal posterior standard deviation vs the frequentist standard error estimated from the .cor file. Large differences often indicate issues with one estimation method.

Value

Invisibly returns data.frame with parameter name and estimated uncertainties.

Examples

```
fit <- readRDS(system.file('examples', 'fit.RDS', package='adnuts'))
x <- plot_uncertainties(fit, plot=FALSE)
head(x)
```

print.adfit *Print summary of adfit object*

Description

Print summary of adfit object

Usage

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

Arguments

x	Fitted object from sample_admb
...	Ignored

Value

Summary printed to console

sample_admb *Deprecated version of wrapper function. Use sample_nuts or sample_rwm instead.*

Description

Deprecated version of wrapper function. Use sample_nuts or sample_rwm instead.

Usage

```
sample_admb(
  model,
  path = getwd(),
  iter = 2000,
  init = NULL,
  chains = 3,
  warmup = NULL,
  seeds = NULL,
  thin = 1,
  mceval = FALSE,
  duration = NULL,
  parallel = FALSE,
  cores = NULL,
  control = NULL,
```

```
  skip_optimization = TRUE,
  algorithm = "NUTS",
  skip_monitor = FALSE,
  skip_unbounded = TRUE,
  admb_args = NULL
)
```

Arguments

model	Name of model (i.e., 'model' for model.tpl). For non-Windows systems this will automatically be converted to './model' internally. For Windows, long file names are sometimes shortened from e.g., 'long_model_filename' to 'LONG_~1'. This should work, but will throw warnings. Please shorten the model name. See https://en.wikipedia.org/wiki/8.3_filename.
path	Path to model executable. Defaults to working directory. Often best to have model files in a separate subdirectory, particularly for parallel.
iter	The number of samples to draw.
init	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., `obj$par`) for all chains.
chains	The number of chains to run.
warmup	The number of warmup iterations.
seeds	A vector of seeds, one for each chain.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
mceval	Whether to run the model with `-mceval` on samples from merged chains.
duration	The number of minutes after which the model will quit running.
parallel	A deprecated argument, use cores=1 for serial execution or cores>1 for parallel (default is to parallel with cores equal to the available-1)
cores	The number of cores to use for parallel execution. Default is number available in the system minus 1. If `cores=1`, serial execution occurs (even if `chains>1`), otherwise parallel execution via package snowfall is used. For slow analyses it is recommended to set `chains<=cores` so each core needs to run only a single chain.
control	A list to control the sampler. See details for further use.
skip_optimization	Whether to run the optimizer before running MCMC. This is rarely need as it is better to run it once before to get the covariance matrix, or the estimates are not needed with adaptive NUTS.
algorithm	The algorithm to use, one of "NUTS" or "RWM"
skip_monitor	Whether to skip calculating diagnostics (effective sample size, Rhat) via the `rstan::monitor` function. This can be slow for models with high dimension or many iterations. The result is used in plots and summaries so it is recommended to turn on. If model run with `skip_monitor=FALSE` you can recreate it post-hoc by setting `fit$monitor=rstan::monitor(fit$samples,fit$warmup, print=FALSE)`.

skip_unbounded	Whether to skip returning the unbounded version of the posterior samples in addition to the bounded ones. It may be advisable to set to FALSE for very large models to save space.
admb_args	A character string which gets passed to the command line, allowing finer control

Warning

This is deprecated and will cease to exist in future releases

---

sample_inits *Function to generate random initial values from a previous fit using adnuts*

Description

Function to generate random initial values from a previous fit using adnuts

Usage

```
sample_inits(fit, chains)
```

Arguments

fit	An outputted list from sample_admb
chains	The number of chains for the subsequent run, which determines the number to return.

Value

A list of lists which can be passed back into sample_admb.

---

sample_nuts *Bayesian inference of an ADMB model using the no-U-turn sampler (NUTS) or random walk Metropolis (RWM) algorithms.*

Description

Draw Bayesian posterior samples from an AD Model Builder (ADMB) model using an MCMC algorithm. ‘sample_nuts‘ and ‘sample_rwm‘ generates posterior samples from which inference can be made.

Usage

```
sample_nuts(
  model,
  path = getwd(),
  iter = 2000,
  init = NULL,
  chains = 3,
  warmup = NULL,
  seeds = NULL,
  thin = 1,
  mceval = FALSE,
  duration = NULL,
  parallel = FALSE,
  cores = NULL,
  control = NULL,
  skip_optimization = TRUE,
  verbose = TRUE,
  skip_monitor = FALSE,
  skip_unbounded = TRUE,
  admb_args = NULL,
  extra.args = NULL
)

sample_rwm(
  model,
  path = getwd(),
  iter = 2000,
  init = NULL,
  chains = 3,
  warmup = NULL,
  seeds = NULL,
  thin = 1,
  mceval = FALSE,
  duration = NULL,
  parallel = FALSE,
  cores = NULL,
  control = NULL,
  skip_optimization = TRUE,
  verbose = TRUE,
  skip_monitor = FALSE,
  skip_unbounded = TRUE,
  admb_args = NULL,
  extra.args = NULL
)
```

Arguments

model	Name of model (i.e., 'model' for model.tpl). For non-Windows systems this will automatically be converted to './model' internally. For Windows, long file names are sometimes shortened from e.g., 'long_model_filename' to 'LONG_~1'. This should work, but will throw warnings. Please shorten the model name. See https://en.wikipedia.org/wiki/8.3_filename.
path	Path to model executable. Defaults to working directory. Often best to have model files in a separate subdirectory, particularly for parallel.
iter	The number of samples to draw.
init	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., `obj$par`) for all chains.
chains	The number of chains to run.
warmup	The number of warmup iterations.
seeds	A vector of seeds, one for each chain.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
mceval	Whether to run the model with `-mceval` on samples from merged chains.
duration	The number of minutes after which the model will quit running.
parallel	A deprecated argument, use cores=1 for serial execution or cores>1 for parallel (default is to parallel with cores equal to the available-1)
cores	The number of cores to use for parallel execution. Default is number available in the system minus 1. If `cores=1`, serial execution occurs (even if `chains>1`), otherwise parallel execution via package snowfall is used. For slow analyses it is recommended to set `chains<=cores` so each core needs to run only a single chain.
control	A list to control the sampler. See details for further use.
skip_optimization	Whether to run the optimizer before running MCMC. This is rarely need as it is better to run it once before to get the covariance matrix, or the estimates are not needed with adaptive NUTS.
verbose	Flag whether to show console output (default) or suppress it completely except for warnings and errors. Works for serial or parallel execution.
skip_monitor	Whether to skip calculating diagnostics (effective sample size, Rhat) via the `rstan::monitor` function. This can be slow for models with high dimension or many iterations. The result is used in plots and summaries so it is recommended to turn on. If model run with `skip_monitor=FALSE` you can recreate it post-hoc by setting `fit$monitor=rstan::monitor(fit$samples,fit$warmup, print=FALSE)`.
skip_unbounded	Whether to skip returning the unbounded version of the posterior samples in addition to the bounded ones. It may be advisable to set to FALSE for very large models to save space.
admb_args	A character string which gets passed to the command line, allowing finer control
extra.args	Deprecated, use a `admb_args` instead.

Details

Adaptation schemes are used with NUTS so specifying tuning parameters is not necessary. See vignette for options for adaptation of step size and mass matrix. The RWM algorithm provides no new functionality not available from previous versions of ADMB. However, 'sample_rwm' has an improved console output, is setup for parallel execution, and a smooth workflow for diagnostics.

Parallel chains will be run if argument 'cores' is greater than one. This entails copying the folder, and starting a new R session to run that chain, which are then merged back together. Note that console output is inconsistent when using parallel, and may not show. On Windows the R terminal shows output live, but the GUI does not. RStudio is a special case and will not show live, and instead is captured and returned at the end. It is strongly recommended to start with serial execution as debugging parallel chains is very difficult.

Note that the algorithm code is in the ADMB source code, and 'adnuts' provides a wrapper for it. The command line arguments are returned and can be examined by the user. See vignette for more information.

This function implements algorithm 6 of Hoffman and Gelman (2014), and loosely follows package `rstan`. The step size can be adapted or specified manually. The metric (i.e., mass matrix) can be unit diagonal, adapted diagonal (default and recommended), a dense matrix specified by the user, or an adapted dense matrix. Further control of algorithms can be specified with the `control` argument. Elements are:

adapt_delta The target acceptance rate. D

metric The mass metric to use. Options are: "unit" for a unit diagonal matrix; `NULL` to estimate a diagonal matrix during warmup; a matrix to be used directly (in untransformed space).

adapt_delta Whether adaptation of step size is turned on.

adapt_mass Whether adaptation of mass matrix is turned on. Currently only allowed for diagonal metric.

adapt_mass_dense Whether dense adaptation of mass matrix is turned on.

max_treedepth Maximum treedepth for the NUTS algorithm.

stepsize The stepsize for the NUTS algorithm. If `NULL` it will be adapted during warmup.

adapt_init_buffer The initial buffer size during mass matrix adaptation where sample information is not used (default 50)

adapt_term_buffer The terminal buffer size (default 75) during mass matrix adaptation (final fast phase)

adapt_window The initial size of the mass matrix adaptation window, which gets doubled each time thereafter.

refresh The rate at which to refresh progress to the console. Defaults to even 10 progress updates.

The adaptation scheme (step size and mass matrix) is based heavily on those by the software Stan, and more details can be found in that documentation and this vignette.

Warning

The user is responsible for specifying the model properly (priors, starting values, desired parameters fixed, etc.), as well as assessing the convergence and validity of the resulting samples (e.g., through the `coda` package), or with function `launch_shinytmb` before making inference. Specifically, priors must be specified in the template file for each parameter. Unspecified priors will be implicitly uniform.

Author(s)

Cole Monnahan

Examples

```
## Not run:
## This is the packaged simple regression model
path.simple <- system.file('examples', 'simple', package='adnuts')
## It is best to have your ADMB files in a separate folder and provide that
## path, so make a copy of the model folder locally.
path <- 'simple'
dir.create(path)
trash <- file.copy(from=list.files(path.simple, full.names=TRUE), to=path)
## Compile and run model
oldwd <- getwd()
setwd(path)
system('admb simple.tpl')
system('simple')
setwd('..')
init <- function() rnorm(2)
## Run NUTS with defaults
fit <- sample_nuts(model='simple', init=init, path=path)
unlink(path, TRUE) # cleanup folder
setwd(oldwd)

## End(Not run)
```

---

sample_tmb *Bayesian inference of a TMB model using the no-U-turn sampler.*

Description

Draw Bayesian posterior samples from a Template Model Builder (TMB) model using an MCMC algorithm. This function generates posterior samples from which inference can be made. Adaptation schemes are used so specification tuning parameters are not necessary, and parallel execution reduces overall run time.

Usage

```
sample_tmb(
  obj,
  iter = 2000,
  init,
  chains = 3,
  seeds = NULL,
  warmup = floor(iter/2),
  lower = NULL,
```

```
  upper = NULL,
  thin = 1,
  parallel = FALSE,
  cores = NULL,
  path = NULL,
  algorithm = "NUTS",
  laplace = FALSE,
  control = NULL,
  ...
)
```

Arguments

obj	A TMB model object.
iter	The number of samples to draw.
init	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., obj$par) for all chains.
chains	The number of chains to run.
seeds	A vector of seeds, one for each chain.
warmup	The number of warmup iterations.
lower	A vector of lower bounds for parameters. Allowed values are -Inf and numeric.
upper	A vector of upper bounds for parameters. Allowed values are Inf and numeric.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
parallel	A deprecated argument, use cores=1 for serial execution or cores>1 for parallel (default is to parallel with cores equal to the available-1)
cores	The number of cores to use for parallel execution. Default is number available in the system minus 1. If cores=1, serial execution occurs (even if chains>1), otherwise parallel execution via package snowfall is used. For slow analyses it is recommended to set chains<=cores so each core needs to run only a single chain.
path	Path to model executable. Defaults to working directory. Often best to have model files in a separate subdirectory, particularly for parallel.
algorithm	The algorithm to use. NUTS is the default and recommended one, but "RWM" for the random walk Metropolis sampler and "HMC" for the static HMC sampler are available. These last two are deprecated but may be of use in some situations. These algorithms require different arguments; see their help files for more information.
laplace	Whether to use the Laplace approximation if some parameters are declared as random. Default is to turn off this functionality and integrate across all parameters with MCMC.
control	A list to control the sampler. See details for further use.
...	Further arguments to be passed to samplers

Details

This function implements algorithm 6 of Hoffman and Gelman (2014), and loosely follows package `rstan`. The step size can be adapted or specified manually. The metric (i.e., mass matrix) can be unit diagonal, adapted diagonal (default and recommended), or a dense matrix specified by the user. Further control of algorithms can be specified with the `control` argument. Elements are:

adapt_delta The target acceptance rate.

metric The mass metric to use. Options are: "unit" for a unit diagonal matrix; "diag" to estimate a diagonal matrix during warmup; a matrix to be used directly (in untransformed space).

adapt_engaged Whether adaptation of step size and metric is turned on.

max_treedepth Maximum treedepth for the NUTS algorithm.

stepsize The stepsize for the NUTS algorithm. If `NULL` it will be adapted during warmup.

Value

A list containing the samples, and properties of the sampler useful for diagnosing behavior and efficiency.

Warning

This is deprecated and will cease to exist in future releases

Author(s)

Cole Monnahan

See Also

`extract_samples` to extract samples and `launch_shinytmb` to explore the results graphically which is a wrapper for the `launch_shinystan` function.

Examples

```
## Build a fake TMB object with objective & gradient functions and some
## other flags
## Not run:
f <- function(x, order=0){
  if(order != 1) # negative log density
    -sum(dnorm(x=x, mean=0, sd=1, log=TRUE))
  else x # gradient of negative log density
}
init <- function() rnorm(2)
obj <- list(env=list(DLL='demo', last.par.best=c(x=init()), f=f,
  beSilent=function() NULL))
## Run NUTS for this object
fit <- sample_tmb(obj, iter=1000, chains=3, init=init)
## Check basic diagnostics
mon <- rstan::monitor(fit$samples, print=FALSE)
Rhat <- mon[,"Rhat"]
max(Rhat)
```

```
ess <- mon[, 'n_eff']
min(ess)
## Or do it interactively with ShinyStan
launch_shinytmb(fit)

## End(Not run)
```

`sample_tmb_hmc` *Draw MCMC samples from a model posterior using a static HMC sampler.*

Description

Draw MCMC samples from a model posterior using a static HMC sampler.

Usage

```
sample_tmb_hmc(
  iter,
  fn,
  gr,
  init,
  L,
  eps,
  warmup = floor(iter/2),
  seed = NULL,
  chain = 1,
  thin = 1,
  control = NULL
)
```

Arguments

`iter`	The number of samples to draw.
`fn`	A function that returns the log of the posterior density.
`gr`	A function that returns a vector of gradients of the log of the posterior density (same as `fn`).
`init`	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., `obj$par`) for all chains.
`L`	The number of leapfrog steps to take. The NUTS algorithm does not require this as an input. If `L=1` this function will perform Langevin sampling. In some contexts `L` can roughly be thought of as a thinning rate.

eps	The step size. If a numeric value is passed, it will be used throughout the entire chain. A NULL value will initiate sampler_params of eps using the dual averaging algorithm during the first warmup steps.
warmup	The number of warmup iterations.
seed	The random seed to use.
chain	The chain number, for printing only.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
control	A list to control the sampler. See details for further use.

Details

This function implements algorithm 5 of Hoffman and Gelman (2014), which includes adaptive step sizes (eps) via an algorithm called dual averaging.

Value

A list containing samples ('par') and algorithm details such as step size adaptation and acceptance probabilities per iteration ('sampler_params').

References

- Neal, R. M. (2011). MCMC using Hamiltonian dynamics. Handbook of Markov Chain Monte Carlo.
- Hoffman and Gelman (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. 15:1593-1623.

Hoffman and Gelman (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. 15:1593-1623.

See Also

sample_tmb

sample_tmb

---

sample_tmb_nuts *Draw MCMC samples from a model posterior using the No-U-Turn (NUTS) sampler with dual averaging.*

Description

Draw MCMC samples from a model posterior using the No-U-Turn (NUTS) sampler with dual averaging.

Usage

```
sample_tmb_nuts(
  iter,
  fn,
  gr,
  init,
  warmup = floor(iter/2),
  chain = 1,
  thin = 1,
  seed = NULL,
  control = NULL
)
```

Arguments

iter	The number of samples to draw.
fn	A function that returns the log of the posterior density.
gr	A function that returns a vector of gradients of the log of the posterior density (same as fn).
init	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., obj$par) for all chains.
warmup	The number of warmup iterations.
chain	The chain number, for printing only.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
seed	The random seed to use.
control	A list to control the sampler. See details for further use.

Details

This function implements algorithm 6 of Hoffman and Gelman (2014), which includes adaptive step sizes (eps) via an algorithm called dual averaging. It also includes an adaptation scheme to tune a diagonal mass matrix (metric) during warmup.

These fn and gr functions must have Jacobians already applied if there are transformations used.

References

Hoffman and Gelman (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. 15:1593-1623.

See Also

sample_tmb

`sample_tmb_rwm` *[Deprecated] Draw MCMC samples from a model posterior using a Random Walk Metropolis (RWM) sampler.*

Description

[Deprecated] Draw MCMC samples from a model posterior using a Random Walk Metropolis (RWM) sampler.

Usage

```
sample_tmb_rwm(
  iter,
  fn,
  init,
  alpha = 1,
  chain = 1,
  warmup = floor(iter/2),
  thin = 1,
  seed = NULL,
  control = NULL
)
```

Arguments

`iter`	The number of samples to draw.
`fn`	A function that returns the log of the posterior density.
`init`	A list of lists containing the initial parameter vectors, one for each chain or a function. It is strongly recommended to initialize multiple chains from dispersed points. A of NULL signifies to use the starting values present in the model (i.e., `obj$par`) for all chains.
`alpha`	The amount to scale the proposal, i.e, Xnew=Xcur+alpha*Xproposed where Xproposed is generated from a mean-zero multivariate normal. Varying `alpha` varies the acceptance rate.
`chain`	The chain number, for printing only.
`warmup`	The number of warmup iterations.
`thin`	The thinning rate to apply to samples. Typically not used with NUTS.
`seed`	The random seed to use.
`control`	A list to control the sampler. See details for further use.

Details

This algorithm does not yet contain adaptation of `alpha` so some trial and error may be required for efficient sampling.

Value

A list containing samples and other metadata.

References

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. J Chem Phys. 21:1087-1092.

See Also

sample_tmb

---

summary.adfit *Print summary of object of class adfit*

Description

Print summary of object of class adfit

Usage

```
## S3 method for class 'adfit'
summary(object, ...)
```

Arguments

object	Fitted object from sample_admb
...	Ignored

Value

Summary printed to screen

Index