Package 'adnuts'

February 8, 2018

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

Version 1.0.0

Description Bayesian inference using the no-U-turn (NUTS) algorithm by Hoffman and Gelman (2014) <http://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' ('TMB') models and other special cases. Functionality is similar to 'Stan', and the 'rstan' and 'shinystan' packages are used for diagnostics and inference.

Depends R (>= 3.3.0)

URL https://github.com/colemonnahan/adnuts

BugReports https://github.com/colemonnahan/adnuts/issues

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

ByteCompile true

Suggests snowfall (>= 1.84.6.1), shinystan (>= 2.3.0), R2admb (>= 0.7.15), matrixcalc (>= 1.0.3), TMB (>= 1.7.11), stats, knitr, rmarkdown

Imports ellipse, rstan

VignetteBuilder knitr

NeedsCompilation no

Author Cole Monnahan [aut, cre]

Maintainer Cole Monnahan <monnahc@uw.edu>

Repository CRAN

Date/Publication 2018-02-08 09:49:24 UTC

R topics documented:

`adnuts` *adnuts: No-U-turn sampling for Template Model Builder and AD Model Builder*

Description

Draw Bayesian posterior samples from a TMB or 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**

adnuts R package provides NUTS sampling for two existing software platforms: ADMB (Fournier et al. 2011) and TMB (Kristensen et al. 2017, Kristensen 2017). The specific NUTS capabilities include adaptation of step size and metric (mass matrix), parallel execution, and links to diagnostic and inference tools provided by **rstan** and **shinystan**.

For TMB models, **adnuts** provides NUTS and other MCMC algorithms written in R. These can be used with a TMB model by plugging in the `obj$fn` and `obj$gr` functions from the DLL directly. It is possible to use these functions with models outside TMB, as long as the log density and gradients can be calculated. See `sample_tmb` for more details.

The ADMB implementation is different in that the NUTS code is bundled into the ADMB source itself. 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.

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.

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.

`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` or `sample_tmb`.
`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 `launch_shinytmb` 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_shinytmb and launch_shinyadmb.

Examples

```
fit <- readRDS(system.file('examples', 'fit_tmb.RDS', package='adnuts'))
## Examine how step size and treedepth changes as the mass matrix updates
## during warmup
sp <- extract_sampler_params(fit, inc_warmup=TRUE)
plot(0,0, type='n', xlim=c(0,510), ylim=c(0,3), xlab='Iteration',
     ylab='Step size (eps)')
for(i in 1:3) lines(1:1000, sp[sp$chain==i,4], col=i)
legend('topright', cex=.7, legend=paste("chain1", 1:3), lty=1, col=1:3)
plot(0,0, type='n', xlim=c(0,1000), ylim=c(0,10), xlab='Iteration',
     ylab='Treedepth')
for(i in 1:3) lines(1:1000, sp[sp$chain==i,5], col=i)
legend('topright', cex=.7, legend=paste("chain1", 1:3), lty=1, col=1:3)
```

---

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)
```

Arguments

fit	A list returned by sample_tmb or 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.

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 TMB model
fit <- readRDS(system.file('examples', 'fit_tmb.RDS', package='adnuts'))
post <- extract_samples(fit)
tail(apply(post, 2, median))
```

---

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, diag = c("trace", "acf", "hist"), acf.ylim = c(-1, 1),
  ymult = NULL, axis.col = gray(0.5), pars = NULL, label.cex = 0.5,
  limits = NULL, ...)
```

Arguments

fit	A list as returned by sample_admb.
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 labels
limits	A list containing the ranges for each parameter to use in plotting.
...	Arguments to be passed to plot call in lower diagonal panels

Value

Produces a plot, and returns nothing.

Author(s)

Cole Monnahan

Examples

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

---

sample_admb *Bayesian inference of an ADMB model using the no-U-turn sampler.*

Description

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

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,
  algorithm = "NUTS", ...)
```

Arguments

model	Name of model (i.e., model.tpl)
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 boolean for whether to use parallel cores. The package snowfall is used if TRUE.
cores	The number of cores to use for parallel execution.
control	A list to control the sampler. See details for further use.
algorithm	Which algorithm to use, either "NUTS" or "RWM".
...	Further arguments to be passed to the algorithm. See help files for the samplers for further arguments.

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. 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.

max_treedepth Maximum treedepth for the NUTS algorithm.

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

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_admb(model='simple', init=init, path=path)
unlink(path, TRUE) # cleanup folder
setwd(oldwd)

## End(Not run)
```

---

sample_admb_nuts *Run a single NUTS chain for an ADMB model*

Description

A low level function to run a single chain. Unlikely to be used by a user, instead prefer sample_admb

Usage

```
sample_admb_nuts(path, model, iter = 2000, init = NULL, chain = 1,
  thin = 1, warmup = NULL, seed = NULL, duration = NULL,
  control = NULL, verbose = TRUE, extra.args = NULL)
```

Arguments

path	Path to model executable. Defaults to working directory. Often best to have model files in a separate subdirectory, particularly for parallel.
model	Name of model (i.e., model.tpl)
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.

chain	Chain number, for printing purposes only.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
warmup	The number of warmup iterations.
seed	Random seed to use.
duration	The number of minutes after which the model will quit running.
control	A list to control the sampler. See details for further use.
verbose	Boolean for whether to print ADMB output to console.
extra.args	Character string of extra command line argument to pass to ADMB.

See Also

sample_admb

`sample_admb_rwm` *Run a single random walk Metropolis chain for an ADMB model*

Description

A low level function to run a single chain. Unlikely to be used by a user, instead prefer sample_admb

Usage

```
sample_admb_rwm(path, model, iter = 2000, thin = 1,
  warmup = ceiling(iter/2), init = NULL, chain = 1, seed = NULL,
  control = NULL, verbose = TRUE, extra.args = NULL, duration = NULL)
```

Arguments

path	Path to model executable. Defaults to working directory. Often best to have model files in a separate subdirectory, particularly for parallel.
model	Name of model (i.e., model.tpl)
iter	The number of samples to draw.
thin	The thinning rate to apply to samples. Typically not used with NUTS.
warmup	The number of warmup iterations.
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.
chain	Chain number, for printing purposes only.
seed	Random seed to use.
control	A list to control the sampler. See details for further use.
verbose	Boolean for whether to print ADMB output to console.
extra.args	Character string of extra command line argument to pass to ADMB.
duration	The number of minutes after which the model will quit running.

See Also

sample_admb

---

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 boolean for whether to use parallel cores. The package snowfall is used if TRUE.
cores	The number of cores to use for parallel execution.
path	The path to the TMB DLL. This is only required if using parallel, since each core needs to link to the DLL again.
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 the algorithm. See help files for the samplers for further arguments.

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

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

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
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
## Not run:
  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

Index