Package 'runjags'

March 26, 2013

Version 1.0.0-6

Date 2013-03-26

Title Interface utilities for MCMC models in Just Another Gibbs Sampler (JAGS) using parallel and distributed computing methods

Author Matthew Denwood <matthew.denwood@glasgow.ac.uk>

Maintainer Matthew Denwood <matthew.denwood@glasgow.ac.uk>

Depends R (>= 2.14), coda (>= 0.16-1), lattice (>= 0.20-10), parallel

Imports coda, lattice, parallel, stats, utils

Suggests rjags

SystemRequirements jags (see http://mcmc-jags.sourceforge.net)

Description This package provides high-level interface utilities for JAGS, either running locally (via the rjags package or using multiple cores in parallel) or via distributed computing clusters such as those provided by snow (a Simple Network Of Workstations), Apple Xgrid distributed computing clusters (Mac OS X 10.5-10.7 only), and possibly others via user specified functions. The primary motivation is to facilitate running relatively simple JAGS models to convergence, including evaluating the performance of a model against simulated data,and compatibility with the WinBUGS syntax of model files with data and initial values lists. Runjags interface functions also provide convenience wrappers for automatic control of model convergence assessment and run length diagnostics,calculation of relevant summary statistics, generation of trace and density plots, calculation of DIC, and automatic retrieval of R objects as data and initial values. Running of arbitrary R commands (not involving JAGS) over Xgrid is also supported.

License GPL

URL http://cran.r-project.org/web/packages/runjags/

NeedsCompilation no

Repository CRAN

Date/Publication 2013-03-26 14:43:36

R topics documented:

`ask` *Obtain Input from User With Error Handling*

Description

A simple function to detect input from the user, and keep prompting until a response matching the class of input required is given.

Usage

```
ask(prompt="?", type="logical", bounds=c(-Inf, Inf),
   na.allow=FALSE)
```

Arguments

`prompt`	what text string should be used to prompt the user? (character string)
`type`	the class of object expected to be returned - "logical", "numeric", "integer", "character". If the user input does not match this return, the prompt is repeated
`bounds`	the lower and upper bounds of number to be returned. Ignored if type is "logical" or "character"
`na.allow`	if TRUE, allows the user to input "NA" for any type, which is returned as NA

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

readline,

menu

Examples

```
#  Ask the user if they want to proceed
## Not run:
ask("Do you want to start the program now?", type="logical")

## End(Not run)
```

---

autorun.jags *Run or Extend a User Specified Bayesian MCMC Model in JAGS with Automatically Calculated Run Length and Convergence Diagnostics*

Description

Runs or extends a user specified JAGS (similar to WinBUGS) model from within R, returning an object of class runjags-class. Chain convergence over the first run of the simulation is assessed using the Gelman and Rubin's convergence diagnostic. If necessary, the simulation is extended to improve chain convergence (up to a user-specified maximum time limit), before the required sample size of the Markov chain is calculated using Raftery and Lewis's diagnostic. The simulation is extended to the required sample size dependant on autocorrelation and the number of chains.

This function is provided primarily for automated running of large simulated data studies, and is not a replacement for manually assessing convergence and Monte Carlo error when parameter estimates are being made from real data. For more complex models, the use of run.jags directly with manual assessment of necessary run length may be preferable.

Requires Just Another Gibbs Sampler (JAGS), see http://www-fis.iarc.fr/~martyn/software/jags/.

Usage

```
autorun.jags(model=stop("No model supplied"), monitor = NA, data=NA,
n.chains=NA, inits = NA, startburnin = 5000, startsample = 10000,
datalist=NA, initlist=NA, psrf.target = 1.05, normalise.mcmc = TRUE,
check.stochastic = TRUE, modules=c(""), factories=c(""),
raftery.options = list(), crash.retry=1, summarise = TRUE,
confidence=0.95, plots = summarise, thin.sample = FALSE, jags =
findjags(), silent.jags = FALSE, interactive=FALSE, max.time=Inf,
adaptive=list(type="burnin", length=200), thin = 1, monitor.deviance
```

```
= FALSE, monitor.pd = FALSE, tempdir=TRUE, jags.refresh=0.1,
batch.jags=silent.jags, method=if ('rjags' %in% .packages())
'rjags' else 'interruptible', method.options=list())

autoextend.jags(runjags.object=stop("The output of a runjags function (with class 'runjags') must b
    add.monitor=character(0), drop.monitor=character(0), drop.chain=numeric(0),
combine=length(c(add.monitor,drop.monitor,drop.chain))==0,
startburnin = 0, startsample = 10000, psrf.target = 1.05,
normalise.mcmc = TRUE, check.stochastic = TRUE, raftery.options =
list(), crash.retry=1, summarise = TRUE, confidence=0.95, plots =
summarise, thin.sample = FALSE, jags = findjags(), silent.jags =
FALSE, interactive=FALSE, max.time=Inf, adaptive=list(type='burnin', length=200),
thin = runjags.object$thin, tempdir=TRUE,
jags.refresh=0.1, batch.jags=silent.jags, method=NA,
method.options=NA)
```

Arguments

model	either a relative or absolute path to a textfile (including the file extension) containing a model in the JAGS language and possibly monitored variable names, data and/or initial values, or a character string of the same. No default. The model must be started with the string 'model{' and ended with '}' on new lines. Data must be similarly started with 'data{', monitored variables with 'monitor{', and initial values as 'inits{', and all ended with '}'. If multiple models are found, all but the first one are ignored with a warning. Multiple data blocks and monitor blocks are combined, multiple inits blocks are used for different chains. The model block may also contain automatically generated data and initial values variables using '#data# variable' and '#inits# variable', and more monitored variables using '#monitor# variable'. See read.winbugs for more details. No default.
monitor	a character vector of the names of variables to monitor. The special node names 'deviance', 'pd', 'pd.i', 'popt' and 'dic' are used to monitor these model fit diagnostics (see the JAGS user manual for more information), but with the exception of 'deviance' these monitored nodes won't appear as variables in the summary statistics or plots. Note: multiple chains are required for calculation of 'pd.i', 'pd', 'popt' and 'dic'.
data	either a named list or a character string in the R dump format containing the data. If left as NA, the model will be run without external data.
n.chains	the number of chains to use with the simulation. More chains will improve the sensitivity of the convergence diagnostic, but will cause the simulation to run more slowly (although this may be improved by using a method such as 'parallel' or 'snow'). The minimum (and default) number of chains is 2.
inits	either a character vector with length equal to the number of chains the model will be run using, or a list of named lists representing names and corresponding values of inits for each chain. If a vector, each element of the vector must be a character string in the R dump format representing the initial values for that

chain, or NA. If not all initialising variables are specified, the unspecified variables are sampled from the prior distribution by JAGS. Values left as NA result in all initial values for that chain being sampled from the prior distribution. The special variables '.RNG.seed', '.RNG.name', and '.RNG.state' are allowed for explicit control over random number generators in JAGS. Default NA.

`runjags.object` the model to be extended - the output of a run.jags (or autorun.jags or extend.jags etc) function, with class 'runjags'. No default.

`add.monitor` a character vector of variables to add to the monitored variable list. All previously monitored variables are automatically included - although see the 'drop.monitor' argument. Default no additional monitors.

`drop.monitor` a character vector of previously monitored variables to remove from the monitored variable list for the extended model. Default none.

`drop.chain` a numeric vector of chains to remove from the extended model. Default none.

`combine` a logical flag indicating if results from the new JAGS run should be combined with the previous chains. Default TRUE if not adding or removing variables or chains, and FALSE otherwise.

`startburnin` the number of burnin iterations. Default 0.

`startsample` the total number of samples (including the chains supplied in runjags.object for autoextend.jags) on which to assess convergence. If the runjags.object already contains this number of samples then convergence will be assessed on this object, otherwise the required number of additional samples will be obtained before combining the chains with the old chains. More samples will give a better chance of allowing the chain to converge, but will take longer to achieve. Also controls the length of the pilot chain used to assess the required sampling length. The minimum is 4000 samples, which is the minimum required number of samples for a model with no autocorrelation and good convergence. Default 10000 iterations.

`datalist` an optional named list containing variables used as data, or alternatively a function (with no arguments) that returns a named list. If any variables are specified in the model block using '#data# <variable>', the value for the corresponding named variable is taken from datalist if present (or the result of datalist() if specified as a function which is useful for specifying randomly generated data), or the parent environment, or finally the global environment if not found anywhere else. Ignored if '#data# <variable>' is not used in the model block. Default NA.

`initlist` an optional named list containing variables used as initial values, or alternatively a function (with a single argument representing the chain number) that returns a named list. If any variables are specified in the model block using '#inits# <variable>', the value for the corresponding named variable is taken from initlist if present (or the result of datalist(chain.no) if specified as a function which allows both randomly generated initial values and different values for each chain), or the parent environment, or finally the global environment if not found anywhere else. Ignored if '#inits# <variable>' is not used in the model block. Note: different chains are all given the same starting values if specified as a named list or taken from any envirnoment; if different values are desired for each chain initlist should be specified as a function. Default NA.

`psrf.target` the value of the point estimate for the potential scale reduction factor of the Gelman Rubin statistic below which the chains are deemed to have converged (must be greater than 1). Default 1.05.

`normalise.mcmc` the Gelman Rubin statistic is based on the assumption that the posterior distribution of monitored variables is roughly normal. For very skewed posterior distributions, it may help to log/logit transform the posterior before calculating the Gelman Rubin statistic. If normalise.mcmc == TRUE, the normality of the untransformed and log/logit transformed posteriors are compared for each monitored variable and the least skewed is used to calculate the Gelman Rubin statistic (this may take some time for large numbers of monitored variables). If FALSE, the data are left untransformed (this may give problems calculating the statistic in extreme cases). Default TRUE.

`check.stochastic` non-stochastic monitored variables will cause errors when calculating the Gelman-Rubin statistic, if check.stochastic==TRUE then all monitored variables will be checked to ensure they are stochastic beforehand. This has a small computational cost, which can be avoided by specifying check.stochastic==FALSE. Default TRUE.

`modules` a character vector of external modules to be loaded into JAGS. More than 1 module can be used. Default none.

`factories` a character vector of factory modules to be loaded into JAGS. More than 1 factory can be used. Factories should be provided in the format '<facname>(<factype>)', for example: factories='mix::TemperedMix(sampler)'. Also ensure that any required modules are also specified (in this case 'mix', for example). Default none.

`raftery.options` a named list which is passed as additional arguments to `raftery.diag`. Default none (default arguments to raftery.diag are used).

`crash.retry` the number of times to re-attempt a simulation if the model returns an error. Default 1 retry (simulation will be aborted after the second crash).

`summarise` should summary statistics be automatically calculated for the output chains? Default TRUE.

`confidence` the prob argument to be passed to HPDinterval for calculation of confidence intervals. Default 0.95 (95% confidence intervals).

`plots` should traceplots and density plots be produced for each monitored variable? If TRUE, the object of class 'runjags' returned will include elements 'trace' and 'density' which consist of a list of lattice objects, with a specific print function that can also be accessed using plot(results). The alternative is to use plot(as.mcmc.list(results)) to look at the density and traceplots for each variable using the traditional graphics system. See also `runjags-class`. Default TRUE.

`thin.sample` option to thin the final MCMC chain(s) before calculating summary statistics and returning the chains. Thinning very long chains allows summary statistics to be calculated more quickly. If TRUE, the chain is thinned to as close to a minimum of startsample iterations as possible (i.e. using a thinning interval of floor(chain.length/thin.sample) since the value must be an integer) and any

excess iterations discarded to ensure the chain length matches thin.sample. If FALSE the chains are not thinned. A positive integer can also be specified as the desired chain length after thinning; the chains will be thinned to as close to this minimum value as possible. Default TRUE (thinned chains of length startsample returned). This option does NOT carry out thinning in JAGS, therefore R must have enough available memory to hold the chains BEFORE thinning. To avoid this problem use the 'thin' option instead.

`jags` the system call or path for activating JAGS. Default calls findjags() to attempt to locate JAGS on your system.

`silent.jags` should the JAGS output be suppressed? (logical) If TRUE, no indication of the progress of individual models is supplied. Default FALSE.

`interactive` option to allow the simulation to be interactive, in which case the user is asked if the simulation should be extended when run length and convergence calculations are performed and the extended simulation will take more than 1 minute. The function will wait for a response before extending the simulations. If FALSE, the simulation will be run until the chains have converged or until the next extension would extend the simulation beyond 'max.time'. Default FALSE.

`max.time` the maximum time for which the function is allowed to extend the chains to improve convergence, as a character string including units or as an integer in which case units are taken as seconds. Ignored if interactive==TRUE. If the function thinks that the next simulation extension to improve convergence will result in a total time of greater than max.time, the extension is aborted. The time per iteration is estimated from the first simulation. Acceptable units include 'seconds', 'minutes', 'hours', 'days', 'weeks', or the first letter(s) of each. Default "1hr".

`adaptive` a list of advanced options controlling the length of the adaptive mode of each simulation. Extended simulations do not require an adaptive phase, but JAGS prints a warning if one is not performed. Reduce the length of the adpative phase for very time consuming models. 'type' must be one of 'adaptive' or 'burnin'.

`thin` the thinning interval to be used in JAGS. Increasing the thinning interval may reduce autocorrelation, and therefore reduce the number of samples required, but will increase the time required to run the simulation. Using this option thinning is performed directly in JAGS, rather than on an existing MCMC object as with thin.sample. Default 1.

`monitor.deviance` this argument is deprecated and remains for backwards compatibility only. See the 'monitor' variable.

`monitor.pd` this argument is deprecated and remains for backwards compatibility only. See the 'monitor' variable.

`tempdir` option to use the temporary directory as specified by the system rather than creating files in the working directory. Any files created in the temporary directory are removed when the function exits for any reason. Default TRUE.

`jags.refresh` the refresh interval (in seconds) for monitoring JAGS output using the 'interactive' and 'parallel' methods (see the 'method' argument). Longer refresh intervals will use less processor time. Default 0.1 seconds.

`batch.jags` option to call JAGS in batch mode, rather than using input redirection. On JAGS >= 3.0.0, this suppresses output of the status which may be useful in some situations. Default TRUE if silent.jags is TRUE, or FALSE otherwise.

`method` the method with which to call JAGS; probably a character vector specifying one of 'rjags', 'simple', 'interruptible', 'parallel', or 'snow' (and see also `xgrid.autoextend.jags`). The former runs JAGS using the rjags package, whereas other options do not require the rjags package and call JAGS as an external executable. 'simple' runs JAGS as a foreground process (the default behaviour for runjags < 0.9.6), 'interruptible' allows the JAGS process to be terminated immediately using the interrupt signal, 'parallel' runs each chain as a separate process on a separate core, and 'snow' uses a simple network of workstations (which may be passed into the method.options list as 'cl'). The default for autorun.jags is to use 'rjags' if the 'rjags' package is load()ed, or the 'interactive' method otherwise. The default for the autoextend.jags function is to use the same method as used for the previous JAGS call.

Note that the parallel, snow and bgparallel methods all use separate JAGS instances to speed up execution of models with multiple chains (at the expense of using more RAM), but cannot be used to monitor pd/popt/pd.i (and therefore DIC). If no .RNG.name is specified in the initial values, each chain is assigned a different random number generator (.RNG.name) for up to 4 chains (the number of different RNG available in JAGS), or using the lecuyer module for 5 or more chains (requires rjags to be installed).

Starting with runjags version 1.0.0, there has been an attempt to abstract the method used to call JAGS - as a result a user-specified function may also be passed as the method. This function must call JAGS on a given batch script contained within 'sim' folders in the working directory, and either wait for JAGS to output simulation results to file then return TRUE or return FALSE which is assumed to mean that the JAGS processes are still running, in which case the behaviour is as for 'background'. The return may also be a list including the named element 'complete' which refers to the same thing, as well as other elements that are returned to the top level. If you are interested in developing another method please feel free to contact the package author.

`method.options` an optional named list of argument to be passed to the method function (including a user specified method function). Of the default arguments, only 'nsims' indicating the number of separate simulations (for parallel, snow and bgparallel methods) and 'cl' specifying an existing snow cluster and/or 'remote.jags' specifying the path to JAGS on the remote machines (for the snow method only) can be used. Others are ignored with a warning.

Value

an object of class 'runjags' (see `runjags-class`).

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

`run.jags`,

`read.winbugs`,

xgrid.autoextend.jags,
runjags-class

Examples

```
# run a model to calculate the intercept and slope of the expression
# y = m x + c, assuming normal observation errors for y:

## Not run:
# Simulate the data
N <- 100
X <- 1:N
Y <- rnorm(N, 2*X + 10, 1)

# Model in the JAGS format
model <- "model {
for(i in 1 : N){
Y[i] ~ dnorm(true.y[i], precision);
true.y[i] <- (m * X[i]) + c;
}
m ~ dunif(-1000,1000);
c ~ dunif(-1000,1000);
precision ~ dexp(1);

#data# N, X, Y
}"


# Run the model
results <- autorun.jags(model=model, monitor=c("m", "c", "precision"))

# Analyse traceplots of the results to assess convergence:
plot(results, type="trace", layout=c(3,1))

# Summary of monitored variables:
results


## End(Not run)
```

`combine.mcmc` *Combine Two or More MCMC Objects Into One Longer MCMC Object*

Description

Allows an MCMC object (with 1 or more chains) to be combined with another (or several other) MCMC object(s) representing extensions of the same simulation, to produce one MCMC object that contains the continuous combined Markov chains of the other MCMC objects. Alternatively, a single MCMC list object can be converted into an MCMC object with one chain by combining all chains from the MCMC list.

Usage

```
combine.mcmc(mcmc.objects=list(), thin=1, return.samples=NA,
collapse.chains=if(length(mcmc.objects)==1) TRUE else FALSE, vars=NA)
```

Arguments

mcmc.objects	a list of MCMC or runjags objects, all with the same number of chains and matching variable names, or a single MCMC object/list or runjags object. No default.
thin	an integer to use to thin the (final) MCMC object by, in addition to any thinning already applied to the objects before being passed to combine.mcmc. Ignored if return.samples is specified (!is.na). Default 1 (no additional thinning is performed).
return.samples	the number of samples to return after thinning. The chains will be thinned to as close to this minimum value as possible, and any excess iterations discarded. Supersedes thin if both are specified. Ignored if niter(mcmc.objects) < return.samples. Default NA.
collapse.chains	option to combine all MCMC chains into a single MCMC chain with more iterations. Can be used for combining chains prior to calculating results in order to reduce the Monte Carlo error of estimates. Default TRUE if a single mcmc.object is provided, or FALSE otherwise.
vars	an optional character vector of variable names to extract. If supplied, only variable names in the MCMC object/list supplied with a partial match to anything in 'vars' will be summarised/plotted/extracted. Note that regular expressions are not allowed, but the caret (^) token can be used to specify the match at the start of a variable name, and a quoted vars will be matched exactly. Default NA meaning all variables available are returned.

Value

An MCMC object if collapse.chains==TRUE or a list of MCMC objects is supplied, or an mcmc.list object if given a list of (or single) mcmc.list or runjags objects (and collapse.chains==FALSE)

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

run.jags,

runjags-class

dump.format *Conversion Between a Named List and a Character String in the R Dump Format*

Description

Convert a named list of numeric vector(s) or array(s) of data or initial values to a character string in the correct format to be read by JAGS as either data or initial values, using the run.jags function.

Usage

```
dump.format(namedlist=list(), checkvalid=TRUE)

list.format(data=character(), checkvalid=TRUE)
```

Arguments

namedlist	a named list of numeric or integer (or something that can be coerced to numeric) vectors, matrices or arrays. The name of each list item will be used as the name of the resulting dump.format variables.
data	a character string in the R dump format, such as that produced by dump.format.
checkvalid	option to ensure that the object returned from the function does not contain any values that would be invalid for import into JAGS, such as Inf, -Inf, character or factor values etc.

Details

The 'dump.format' function creates a character string of the supplied variables in the same way that dump() would, except that the result is returned as a character string rather than written to file. Additionally, dump.format() will look for any variable with the name '.RNG.name' and double quote the value if not already double quoted (to ensure compatibility with JAGS).

Value

Either a character string in the R dump format (for dump.format), or a named list (for list.format).

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

run.jags, dump

Examples

```
# A named list:
namedlist1 <- list(N=10, Count=c(4,2,7,0,6,9,1,4,12,1))

# Conver to a character vector:
chardata <- dump.format(namedlist1)

# And back to a named list:
namedlist2 <- list.format(chardata)

# These should be the same:
stopifnot(identical(namedlist1, namedlist2))
```

`findjags` *Attempt to Locate a JAGS Install*

Description

Search the most likely locations for JAGS to be installed on the users system, based on the operating system, and return the most likely path to try. Where multiple installs exist, findjags will attempt to return the path to the install with the highest version number. For Unix systems, calling jags using 'jags' requires the jags binary to be in the search path, which may be specified in your user '.Profile' if necessary (the JAGS executable is also looked for in the default install location of /usr/local/bin/jags if popen support is enabled).

Usage

```
findjags(ostype = .Platform$OS.type,
look_in = if(ostype=="windows") c("/Program Files/","/Windows/Program Files/","C:/Program Files/","
from.variable = ".jagspath")
```

Arguments

`ostype`	the operating system type. There is probably no reason to want to change this...
`look_in`	for Windows only, the path to a folder (or vector of folders) which contains another folder with name containing 'JAGS', where the JAGS executable(s) are to be found. findjags() will attempt to find the highest version, assuming that the version number is somewhere in the file path to the executable (as per default installation).
`from.variable`	a global variable that may contain the path to JAGS. This provides a 'set and forget' solution if you have a non-standard JAGS install, by assigning the variable ".jagspath" (the default argument value) to the path to JAGS in your R profile file (this will be read every time R starts).

Value

A path or command for the most likely location of the desired JAGS executable on the system. On unix this will always be 'jags', on Windows for example "C:/Program Files/JAGS/bin/jags-terminal.exe" or "C:/Program Files/JAGS/JAGS-1.0.0/bin/jags-terminal.exe"

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

testjags,

run.jags

---

new_unique *Create a Unique Filename*

Description

Search the current working directory for a file or directory matching the input name, and if it exists suggest a new name by appending a counter to the input name. Alternatively, the function can ask the user if the existing file should be overwritten, in which case the existing file will be erased if the answer is 'yes'. The function also checks for write access permissions at the current working directory.

Usage

```
new_unique(name = NA, suffix = "", ask = FALSE,
prompt = "A file or directory with this name already exists.  Overwrite?",
touch=FALSE, type='file')
```

Arguments

name	the filename to be used (character string). A vector of character strings is also permissible, in which case they will be pasted together. One or more missing (NA) values can also be used, which will be replaced with a randomly generated 9 character alphanumeric string. Default NA.
suffix	the file extension (including '.') to use (character string). If this does not start with a '.', one will be prepended automatically. Default none.
ask	if a file exists with the input name, should the function ask to overwrite the file? (logical) If FALSE, a new filename is used instead and no files will be over-written. Default FALSE.
prompt	what text string should be used to prompt the user? (character string) Ignored is ask==FALSE. A generic default is supplied.

touch	option to create (touch) the file/folder after generating the unique name, which prevents other processes from sneaking in and creating a file with the same name before the returned filename has had chance to be used. Default FALSE.
type	if touch==TRUE, then type controls if a file or directory is created. One of 'file', 'f', 'directory', or 'd'. Defult 'file'.

Value

A unique filename that is safe to use without fear of destroying existing files

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

ask

Examples

```
#  Create a file name that is unlikely to exist already,
#  with a .R extension.
new_unique(c("new_file", NA), ".R", ask=FALSE)
```

---

read.winbugs *Extract Any Models, Data, Monitored Variables or Initial Values As Character Vectors from a Winbugs Type Textfile*

Description

Read a user specified WinBUGS type textfile or character variable and extract any models, data, monitored variables or initial values as character vectors. Used by (auto)run.jags to interpret the input file(s) or strings.

Usage

```
read.winbugs(path)
```

Arguments

path	either a relative or absolute path to a textfile (including the file extension) containing a model in the JAGS language and possibly monitored variable names, data and/or initial values, or a character string of the same. May also be a vector of paths to different text files, possibly separately containing the model, data and intitial values. No default. The model must be started with the string 'model{' and ended with '}' on new lines. Data must be similarly started with 'data{',

monitored variables with 'monitor{', and initial values as 'inits{', and all ended with '}'. Seperate variables in such blocks must be separated by a line break. If multiple models are found, all but the first one are ignored with a warning. Multiple data blocks and monitor blocks are combined, multiple inits blocks are used for different chains. Monitors may also be given using the phrase '#monitor# variable' within the model block, in which case 'variable' is added to the list of monitored variables found in the monitor block(s). The use of automatically generated data and initial values is also supported using similar syntax, with '#data# variable' for automatically generated data variables or '#inits# variable' for automatically generated initial value variables in which case 'variable' is used as data or initial values with a value taken by `run.jagsfile` from datalist, initlist or R objects as appropriate. '#inits#', '#data#' and '#monitor#' statements can appear on the same line as model code, but no more than one of these statements should be used on the same line. Examples of acceptable model syntax are given below.

Value

A named list of 'model' containing the model description, 'data' containing the data given in the data block(s), 'autodata' containing data variables specified using '#data#' in the model block, 'inits' containing the initial values given in the initial value block(s), 'autoinits' containing initial value variables specified using '#inits#' in the model block, and 'monitor' containing the monitored variables specified in the monitor blocks and by using '#monitor#' within the model block. This function is specified primarily for WinBugs compatibility, so data blocks would normally contain the data in a list format rather than the code format that is allowed in JAGS to perform data transformations (see JAGS manual section 7.0.5). These JAGS format data blocks can be specified, and the function will attempt to differentiate the two types of data from the presence of syntactical cues such as square brackets, for loops, 'list' and .Dim structural assignments. If none of these are found, the data block is assumed to be a WinBugs type data block and is passed to JAGS as data. This behaviour can be over-ridden by inserting '#jagsdata#' or '#bugsdata#' into the data block as appropriate. More than one data block is allowed, and each will be differentiated independently.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

`run.jags`

Examples

```
## Not run:

# ALL SYNTAX GIVEN BELOW IS EQUIVALENT

# Use a modified WinBUGS text file with manual inits and manual data and
```

```
# a seperate monitor block (requires least modification from a WinBUGS
# file).  For compatibility with WinBUGS, the use of list() to enclose
# data and initial values is allowed and ignored, however all seperate
# variables in the data and inits blocks must be seperated with a line
# break (commas or semicolons before linebreaks are ignored).  'data{'
# and 'inits{' must also be added to WinBUGS textfiles so that the
# function can seperate data from initial values.  Iterative loops are
# allowed in data blocks but not in init blocks.  See also the differences
# in JAGS versus WinBUGS syntax in the JAGS help file.

# Contents of a textfile 'mymodel.bug':

model{

for(i in 1:N){
Count[i] ~ dpois(mean)
}
mean ~ dgamma(0.01, 100)
}

data{
list(Count <- c(1,2,3,4,5,6,7,8,9,10),
N <- 10)

}
inits{
list(
mean <- 1)
}

inits{
list(
mean <- 100)
}

monitor{
mean
}

# end text file

read.winbugs('pathtofile/mymodel.bug')


# Use internal character variable, define monitors in the model,
# use autodata and manual initial values:

string <- "
model{

for(i in 1:N){
Count[i] ~ dpois(mean) #data# Count, N
}
```

```
mean ~ dgamma(0.01, 100)
#monitor# mean
}

inits{
mean <- 1
}

inits{
mean <- 100
}
"

read.winbugs(string)


# Use autoinits and a mixture of manual and autodata:
string <- "
model{

for(i in 1:N){
Count[i] ~ dpois(mean) #data# Count
}
mean ~ dgamma(0.01, 100)
#monitor# mean
#inits# mean
}

data{

N <- 10

}
"

read.winbugs(string)


## End(Not run)
```

`run.jags` *Run or Extend a User Specified Bayesian MCMC Model in JAGS from Within R*

Description

Runs or extends a user specified JAGS model from within R, returning an object of class `runjags-class`. Data and initial values can either be supplied in the R dump format (see dump.format() for an easy

way to do this), or as a named list. A character vector of variables to monitor must also be supplied, either to the monitor argument or inside the model code.

Requires Just Another Gibbs Sampler (JAGS), see http://www-fis.iarc.fr/~martyn/software/jags/.

Usage

```
run.jags(model=stop("No model supplied"), monitor = NA, data=NA,
n.chains=NA, inits = NA, burnin = 5000, sample = 10000,
adapt=max(200-burnin, 0), datalist=NA, initlist=NA, jags =
findjags(), silent.jags = FALSE, summarise = TRUE, confidence=0.95,
plots = summarise, psrf.target = 1.05, normalise.mcmc = TRUE,
check.stochastic = TRUE, modules=c(""), factories=c(""), thin = 1,
monitor.deviance = FALSE, monitor.pd = FALSE, monitor.pd.i = FALSE,
monitor.popt = FALSE, check.conv = summarise, keep.jags.files =
FALSE, tempdir=TRUE, jags.refresh=0.1, batch.jags=silent.jags,
method=if ('rjags' %in% .packages()) 'rjags' else 'interruptible',
method.options=list())


extend.jags(runjags.object=stop("The output of a runjags function (with class 'runjags') must be su
    add.monitor=character(0), drop.monitor=character(0), drop.chain=numeric(0),
combine=length(c(add.monitor,drop.monitor,drop.chain))==0, burnin =
0, sample = 10000, adapt=max(200-burnin, 0), jags = findjags(),
silent.jags = FALSE, summarise = TRUE, confidence=0.95, plots =
summarise, psrf.target = 1.05, normalise.mcmc = TRUE,
check.stochastic = TRUE, thin = runjags.object$thin, keep.jags.files
= FALSE, tempdir=TRUE, jags.refresh=0.1, batch.jags=silent.jags,
method=NA, method.options=NA)

results.jags(background.runjags.object=stop("An object produced by a background runjags method must
```

Arguments

model	either a relative or absolute path to a textfile (including the file extension) containing a model in the JAGS language and possibly monitored variable names, data and/or initial values, or a character string of the same. No default. The model must be started with the string 'model{' and ended with '}' on new lines. Data must be similarly started with 'data{', monitored variables with 'monitor{', and initial values as 'inits{', and all ended with '}'. If multiple models are found, all but the first one are ignored with a warning. Multiple data blocks and monitor blocks are combined, multiple inits blocks are used for different chains. The model block may also contain automatically generated data and initial values variables using '#data# variable' and '#inits# variable', and more monitored variables using '#monitor# variable'. See read.winbugs for more information. No default.

`monitor` a character vector of the names of variables to monitor. The special node names 'deviance', 'pd', 'pd.i', 'popt' and 'dic' are used to monitor these model fit diagnostics (see the JAGS user manual for more information), but with the exception of 'deviance' these monitored nodes won't appear as variables in the summary statistics or plots. Note: multiple chains are required for calculation of 'pd.i', 'pd', 'popt' and 'dic'. No default.

`data` a character string in the R dump format (or a named list) containing the data. If left as NA, no external data is used in the model. Default NA.

`n.chains` the number of chains to use with the simulation. More chains will improve the sensitivity of the convergence diagnostic, but will cause the simulation to run more slowly (although this may be improved by using a method such as 'parallel' or 'snow'). The minimum (and default) number of chains is 2.

`inits` either a character vector with length equal to the number of chains the model will be run using, or a list of named lists representing names and corresponding values of inits for each chain. If a vector, each element of the vector must be a character string in the R dump format representing the initial values for that chain, or NA. If not all initialising variables are specified, the unspecified variables are sampled from the prior distribution by JAGS. Values left as NA result in all initial values for that chain being sampled from the prior distribution. The special variables '.RNG.seed', '.RNG.name', and '.RNG.state' are allowed for explicit control over random number generators in JAGS. Default NA.

`runjags.object` the model to be extended - the output of a run.jags (or autorun.jags or extend.jags etc) function, with class 'runjags'. No default.

`background.runjags.object`
the output of a run.jags (or extend.jags) function call using a background JAGS method, with class 'runjags.bginfo'. No default.

`add.monitor` a character vector of variables to add to the monitored variable list. All previously monitored variables are automatically included - although see the 'drop.monitor' argument. Default no additional monitors.

`drop.monitor` a character vector of previously monitored variables to remove from the monitored variable list for the extended model. Default none.

`drop.chain` a numeric vector of chains to remove from the extended model. Default none.

`combine` a logical flag indicating if results from the new JAGS run should be combined with the previous chains. Default TRUE if not adding or removing variables or chains, and FALSE otherwise.

`burnin` the number of burnin iterations (not sampled) to use (numeric). Default 5000 iterations.

`sample` the number of sampling iterations to use (numeric). Default 10000 iterations.

`adapt` advanced option to control the length of the adaptive phase directly, which is otherwise half the length of the burnin period. Default is 0, unless burnin is less than 200 in which case 100 adapitve iterations are used.

`datalist` an optional named list containing variables used as data, or alternatively a function (with no arguments) that returns a named list. If any variables are specified in the model block using '#data# variable', the value for the corresponding

named variable is taken from datalist if present (or the result of datalist() if specified as a function which is useful for specifying randomly generated data), or the parent environment, or finally the global environment if not found anywhere else. Ignored if '#data# variable' is not used in the model block. Default NA.

`initlist` an optional named list containing variables used as initial values, or alternatively a function (with a single argument representing the chain number) that returns a named list. If any variables are specified in the model block using '#inits# variable', the value for the corresponding named variable is taken from initlist if present (or the result of datalist(chain.no) if specified as a function which allows both randomly generated initial values and different values for each chain), or the parent environment, or finally the global environment if not found anywhere else. Ignored if '#inits# variable' is not used in the model block. Note: different chains are all given the same starting values if specified as a named list or taken from any envirnoment; if different values are desired for each chain initlist should be specified as a function. Default NA.

`jags` the system call or path for activating JAGS. Default calls findjags() to attempt to locate JAGS on your system.

`silent.jags` should the JAGS output be suppressed? (logical) If TRUE, no indication of the progress of individual models is supplied. Default FALSE.

`summarise` should summary statistics be assessed after the model has completed? Default TRUE.

`confidence` the prob argument to be passed to HPDinterval for calculation of confidence intervals. Default 0.95 (95% confidence intervals).

`plots` should traceplots and density plots be produced for each monitored variable? If TRUE, the returned list will include elements 'trace' and 'density' which consist of a list of lattice objects accessible using the plot method for the `runjags-class`. The alternative is to use (for example) plot(as.mcmc.list(results)) to look at the density and traceplots for each variable using the traditional graphics system. Default follows the summarise argument (which is required to be TRUE for plots to be produced).

`psrf.target` the value of the point estimate for the potential scale reduction factor of the Gelman Rubin statistic below which the chains are deemed to have converged (must be greater than 1). Ignored if check.conv==FALSE. Default 1.05.

`normalise.mcmc` the Gelman Rubin statistic is based on the assumption that the posterior distribution of monitored variables is roughly normal. For very skewed posterior distributions, it may help to log/logit transform the posterior before calculating the Gelman Rubin statistic. If normalise.mcmc == TRUE, the normality of the untransformed and log/logit transformed posteriors are compared for each monitored variable and the least skewed is used to calculate the Gelman Rubin statistic. If FALSE, the data are left untransformed (this may give problems calculating the statistic in extreme cases). Ignored if check.conv==FALSE. Default TRUE.

`check.stochastic`

non-stochastic monitored variables will cause errors when calculating the Gelman-Rubin statistic, if check.stochastic==TRUE then all monitored variables will be checked to ensure they are stochastic beforehand. This has a computational cost, and can be bypassed if check.stochastic==FALSE. Default TRUE.

`modules` external modules to be loaded into JAGS. More than 1 module can be used. Default none.

`factories` factory modules to be loaded into JAGS. More than 1 factory can be used. Factories should be in the format '<facname>(<factype>)', for example: factories='mix::TemperedMix(sampler)'. Default none.

`thin` the thinning interval to be used in JAGS. Increasing the thinning interval may reduce autocorrelation, and therefore reduce the number of samples required, but will increase the time required to run the simulation. Default 1.

`monitor.deviance` this argument is deprecated and remains for backwards compatibility only. See the 'monitor' variable.

`monitor.pd` this argument is deprecated and remains for backwards compatibility only. See the 'monitor' variable.

`monitor.pd.i` this argument is deprecated and remains for backwards compatibility only. See the 'monitor' variable.

`monitor.popt` this argument is deprecated and remains for backwards compatibility only. See the 'monitor' variable.

`check.conv` this argument is deprecated and remains for backwards compatibility only. See the 'summarise' variable.

`keep.jags.files` option to keep the folder with files needed to call JAGS, rather than deleting it. May be useful for attempting to bug fix models. A character string can also provided, in which case this folder name will be used instead of the default (existing folders will NOT be over-written). Default FALSE.

`tempdir` option to use the temporary directory as specified by the system rather than creating files in the working directory. If keep.jags.files==TRUE then the folder is copied to the working directory after the job has finished (with a unique folder name based on 'runjagsfiles'). Any files created in the temporary directory are removed when the function exits for any reason. Default TRUE.

`jags.refresh` the refresh interval (in seconds) for monitoring JAGS output using the 'interactive' and 'parallel' methods (see the 'method' argument). Longer refresh intervals will use less processor time. Default 0.1 seconds.

`batch.jags` option to call JAGS in batch mode, rather than using input redirection. On JAGS >= 3.0.0, this suppresses output of the status which may be useful in some situations. Default TRUE if silent.jags is TRUE, or FALSE otherwise.

`method` the method with which to call JAGS; probably a character vector specifying one of 'rjags', 'simple', 'interruptible', 'parallel', 'background', 'bgparallel' or 'snow' (and see also `xgrid.autoextend.jags`). The former runs JAGS using the rjags package, whereas other options do not require the rjags package and call JAGS as an external executable. 'simple' runs JAGS as a foreground process (the default behaviour for runjags < 0.9.6), 'interruptible' allows the JAGS process to be terminated immediately using the interrupt signal, 'parallel' runs each chain as a separate process on a separate core, 'snow' uses a simple network of workstations (which may be passed into the method.options list as 'cl'), and 'background' & 'bgparallel' starts JAGS as one or more background processes

and returns the information needed to be passed to `results.jags` to retrieve the simulations when they have finished. The default for run.jags is to use 'rjags' if the 'rjags' package is load()ed, or the 'interactive' method otherwise. The default for the extend.jags function is to use the same method as used for the previous JAGS call.

Note that the parallel, snow and bgparallel methods all use separate JAGS instances to speed up execution of models with multiple chains (at the expense of using more RAM), but cannot be used to monitor pd/popt/pd.i (and therefore DIC). Each chain is specified using a different random number generator (.RNG.name) for up to 4 chains (the number of different RNG available in JAGS), unless .RNG.name is specified in the initial values. Because each chain uses a separate JAGS instance, JAGS has no way of ensuring independence between multiple chains using the same random number generator (as would normally be done when calling a single JAGS instance with multiple chains). Using more than 4 chains with one of these methods without the use of new RNG factories may therefore produce dependence between chains, and is not recommended (a warning is given if trying to do so).

Starting with runjags version 1.0.0, there has been an attempt to abstract the method used to call JAGS - as a result a user-specified function may also be passed as the method. This function must call JAGS on a given batch script contained within 'sim' folders in the working directory, and either wait for JAGS to output simulation results to file then return TRUE or return FALSE which is assumed to mean that the JAGS processes are still running, in which case the behaviour is as for 'background'. The return may also be a list including the named element 'complete' which refers to the same thing, as well as other elements that are returned to the top level. If you are interested in developing another method please feel free to contact the package author.

`method.options` an optional named list of argument to be passed to the method function (including a user specified method function). Of the default arguments, only 'nsims' indicating the number of separate simulations (for parallel, snow and bgparallel methods) and 'cl' specifying an existing snow cluster and/or 'remote.jags' specifying the path to JAGS on the remote machines (for the snow method only) can be used. Others are ignored with a warning.

Value

Usually an object of class 'runjags', or an object of class 'runjags.bginfo' for background methods (see `runjags-class`).

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

`runjags-class`

`autorun.jags`,

`xgrid.run.jags`,

combine.mcmc,

testjags,

dump.format

Examples

```
# run a model to calculate the intercept and slope of the expression
# y = m x + c, assuming normal observation errors for y:

## Not run:

# Simulate the data
X <- 1:100
Y <- rnorm(length(X), 2*X + 10, 1)

# Model in the JAGS format
model <- "model {
for(i in 1 : N){
Y[i] ~ dnorm(true.y[i], precision);
true.y[i] <- (m * X[i]) + c;
}
m ~ dunif(-1000,1000);
c ~ dunif(-1000,1000);
precision ~ dexp(1);
}"

# Use dump.format to convert the data and initial values files
# into the R dump format, with explicit control over the random
# number generator used for each chain (optional):
data <- dump.format(list(X=X, Y=Y, N=length(X)))
inits1 <- dump.format(list(m=1, c=1, precision=1,
.RNG.name="base::Super-Duper", .RNG.seed=1))
inits2 <- dump.format(list(m=0.1, c=10, precision=1,
.RNG.name="base::Wichmann-Hill", .RNG.seed=2))

# Run the model and produce plots
results <- run.jags(model=model, monitor=c("m", "c", "precision"),
data=data, n.chains=2, inits=c(inits1,inits2), plots = TRUE)

# Plot the monitored variables:
plot(results)

# Look at the summary statistics:
print(results)

# Extract only the coefficient as an mcmc.list object:
coeff <- as.mcmc.list(results,vars="m")


## End(Not run)
```

```
# The same model but using embedded shortcuts to specify data, inits and monitors:

## Not run:

# Model in the JAGS format

model <- "model {
for(i in 1 : N){ #data# N
Y[i] ~ dnorm(true.y[i], precision); #data# Y
true.y[i] <- (m * X[i]) + c; #data# X
}
m ~ dunif(-1000,1000); #inits# m
c ~ dunif(-1000,1000);
precision ~ dexp(1);
#monitor# m, c, precision
}"

# Simulate the data
X <- 1:100
Y <- rnorm(length(X), 2*X + 10, 1)
N <- length(X)

initfunction <- function(chain) return(switch(chain,
"1"=list(m=-10), "2"=list(m=10)))

results <- run.jags(model, n.chains=2, initlist=initfunction)

# Look at a traceplot of the intercept and slope on a 2x1 grid:
plot(results,type="trace",vars=c("m","^c"),layout=c(2,1))


## End(Not run)
```

`run.jags.study` *Run an MCMC Model in JAGS Using Multiple Simulated Datasets*

Description

This function can be used to fit a user specified JAGS model to multiple datasets with automatic control of run length and convergence, over a distributed computing cluster such as that provided by snow. The results for monitored variables are compared to the target values provided and a summary of the model performance is returned. This may be used to facilitate model validation using simulated data, or to assess model fit using a 'drop-k' type cross validation study where one or more data points are removed in turn and the model's ability to predict that datapoint is assessed.

Usage

```
run.jags.study(simulations, model=NULL, datafunction=NULL,
targets=list(), confidence=0.95, record.chains=FALSE,
runjags.options=list(), cat.progress=FALSE, test=TRUE,
parallel.method=parLapply, ...)
```

Arguments

simulations	the number of datasets to run the model on
model	the JAGS model to use, in the same format as would be specified to run.jags
datafunction	an optional function that will be used to specify the data. If provided, this must take exactly one argument, representing the simulation number, and return either a named list or character vector in the R dump format containing the data specific to that simulation. It is possible to specify any data that does not change for each simulation using a #data# <variable> tag in the model code. If a datafunction is not provided, the data will be the same for all simulations (and a warning will be printed).
targets	a named list of variables (which can include vectors/arrays) with values to which the model outputs are compared (if stochastic). The target variable names are also automatically included as monitored variables.
confidence	a probability (or vector of probabilities) to use when calculating the proportion of credible intervals containing the true target value. Default 95% CI.
record.chains	option to return the full runjags objects returned from each simulation as a list item named 'runjags'. Default FALSE.
runjags.options	a named list of options to be passed to the underlying autorun.jags function used to run the models.
cat.progress	option to print a message when individual simulations have finished running. This is available for use with lapply, but messages will not be printed for some parallel methods (such as the default parLapply). Default FALSE.
test	option to test the model compilation on a single (randomly chosen) dataset, to ensure that the model compiles before calling the parallel method. Default TRUE.
parallel.method	a function that will be used to call the repeated simulations. This must take the first two arguments 'X' and 'FUN' as for lapply, with other optional arguments passed through from the parent function call. Default uses parLapply, but lapply or mclapply could also be used.
...	optional arguments to be passed directly to the parallel method function, such as 'cl' in the case of parLapply.

Value

An object of class runjags.study-class, containing a summary of the performance of the model with regards to the target variables specified. If record.chains==TRUE, an element named 'runjags' containing a list of all the runjags objects returned will also be present.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

run.jags, runjags.study-class

Examples

```
## Not run:

# Perform a drop-1 validation study for a simple model:

themodel <- "
model{

for(i in 1:N){
Y[i] ~ dnorm(true.y[i], precision)
true.y[i] <- (m * X[i]) + c
}
m ~ dunif(-1000,1000)
c ~ dunif(-1000,1000)
precision ~ dexp(1)

#data# N, X
#inits# m, c, precision
}"

# Simulate the data
N <- 20
X <- 1:N
Y <- rnorm(length(X), 2*X + 1, 1)

# Some initial values to use for 2 chains:
m <- list(-10,10)
c <- list(10,-10)
precision <- list(0.01,100)

# A simple function that removes (over-writes with NA) one datapoint at a time:
datafun <- function(s){
simdata <- Y
simdata[s] <- NA
return(list(Y=simdata))
}
```

```
# Set up a cluster to use with the parLapply method:
cl <- makeCluster(20)

# Call the 20 simulations over the snow cluster:
results <- run.jags.study(simulations=20, model=themodel, datafunction=datafun, targets=list(Y=Y, m=2, c=1), r

# Examine the results:

results



## End(Not run)
```

`run.jagsfile` *Deprecated functions*

Description

These functions are deprecated and only remain for backwards compatibility. Please use the run.jags or autorun.jags functions directly, with the model argument replacing path.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

autorun.jags,

run.jags,

xgrid.run.jags,

read.winbugs

`runjags` *Interface utilities for Just Another Gibbs Sampler (JAGS) using parallel and distributed computing*

Description

This package provides high-level interface utilities for JAGS, either running locally (via the rjags package or using multiple cores in parallel) or via distributed computing clusters such as those provided by snow (a Simple Network Of Workstations), Apple Xgrid distributied computing clusters (Mac OS X 10.5-10.7 only), and possibly others via user specified functions. The primary motivation is to facilitate running relatively simple JAGS models to convergence, including evaluating the performance of a model against simulated data, and compatibility with the WinBUGS syntax of model files with data and initial values lists. Runjags interface functions also provide convenience wrappers for automatic control of model convergence assessment and run length diagnostics, calculation of relevant summary statistics, generation of trace and density plots, calculation of DIC, and automatic retrieval of R objects as data and initial values. Running of arbitrary R commands (not involving JAGS) over Xgrid is also supported. Requires Just Another Gibbs Sampler (JAGS) for most functions, see: http://www-fis.iarc.fr/~martyn/software/jags/

Details

JAGS is a program which allows analysis of Bayesian models using Markov chain Monte Carlo (MCMC) simulation, and was developed by Martyn Plummer to be an alternative to BUGS that ran on UNIX systems as well as Windows systems (see: http://www-fis.iarc.fr/~martyn/software/jags/ for more information). The R package rjags is a native R interface to the JAGS library, and allows a greater level of control for compiled models, which may be more useful for model development. This package was intended to provide additional functions to help automate the process of running models, including interpretation of WinBUGS type text files including data and initial values, automated convergence diagnostics, automated collation and plotting of results, and convinience wrappers for running models (either individually or for multiple data sets) over distributed computing cluster such as those provided by snow and Apple's (now discontinued) Xgrid. The package also includes functions for running any other user specified R code over Xgrid distributed computing clusters from within R (requires Mac OS X and access to an Xgrid system).

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

run.jags and extend.jags for basic model runs

autorun.jags and autoextend.jags for automated running of models to convergence and automated calculation of necessary sample sizes

runjags-class for S3 methods relating to runjags objects, incluing conversion to/from jags objects (for compatibility with the rjags package)

read.winbugs for translation of WinBUGS text files into JAGS compatible model, data and initial values files

combine.mcmc and dump.format for MCMC related tools

timestring, new_unique and ask for more general tools

xgrid.run.jags and xgrid.submit.jags for use of Xgrid clusters to run JAGS models remotely

xgrid.run and xgrid.submit for use of Xgrid clusters for remote execution of user specified R code

jags.model in the rjags package for fine control over the JAGS libraries

---

runjags-class *The 'runjags' class (and associated classes) and available S3 methods*

Description

Objects of class 'runjags' are produced by the run.jags/extend.jags/autorun.jags etc functions, and can be passed to extend.jags and autoextend.jags to extend the simulation. They also have a few specific S3 methods for print, plot and extraction of the MCMC objects contained within the runjags object.

Objects of class 'runjags.study' are produced by the run.jags.study function, and class 'runjags.bginfo' represents a JAGS model being run using a background method and can be passed to the results.jags function to retrieve (a copy of this object will also have been saved to 'jagsinfo.Rsave' in the working directory of the background JAGS call in case the returned object is not saved by the user).

These functions provide print and plot methods, and conversion facilities to/from MCMC objects and objects of class 'jags' for compatibility with the rjags package.

The 'failedjags' environment is used to store JAGS model/data/initial value files from failed simulations for inspection by the user.

Usage

```
## S3 method for class 'runjags'
print(x, vars=NA, digits = 5, ...)
## S3 method for class 'runjags'
plot(x,vars=NA, layout=NA, newwindows=NA,
file="", type="all", ...)

## S3 method for class 'runjags.model'
print(x, linenumbers=TRUE, ...)
## S3 method for class 'runjags.data'
print(x, linenumbers=TRUE, ...)
## S3 method for class 'runjags.inits'
print(x, linenumbers=TRUE, ...)

## S3 method for class 'runjags.study'
print(x,...)
## S3 method for class 'runjags.bginfo'
print(x, ...)

## S3 method for class 'runjags'
as.mcmc(x)
## S3 method for class 'runjags'
as.mcmc.list(x, vars=NA, ...)
```

```
## S3 method for class 'runjags'
as.jags(x, ...)
## S3 method for class 'jags'
as.runjags(x, monitor = stop("No monitored variables supplied"),
modules=c(""), factories=c(""), check=TRUE, jags = findjags(), ...)
```

Arguments

x	an object of class 'runjags' (as returned by the run.jags or autorun.jags functions), or for the as.runjags method an object of class 'jags' (as returned by the jags.model function in the rjags package).
vars	an optional character vector of variable names to extract. If supplied, only variable names in the object supplied with a partial match to anything in 'vars' will be summarised/plotted/extracted. Note that regular expressions are not allowed, but the caret (^) token can be used to specify the match at the start of a variable name, and a quoted vars will be matched exactly. Default NA meaning all variables available are returned.
digits	the number of significant digits to display for tabulated statistics. Default 5.
layout	a numeric vector of length 2 representing the number of rows and columns to produce plots in. Default 1 plot per page for trace and density options, or a single row of 2 plots if plotting both types.
newwindows	if there are a greater number of variables than will fit on one page, should new graphics windows be created for each plot or the exisiting device used for all plots? Ignored if writing plots to file. Default TRUE on Mac/Windows GUI systems, and FALSE otherwise.
file	an optional character string representing a filename to save plots to (as a PDF) rather than using the default graphics device. Default "" (ie don't write to file).
type	option to produce 'trace' plots, 'density' plots, a 'crosscorr' plot or 'all'. Cases are ignored and partial matching is used; the argument may also be of length >1. Default 'all'.
linenumbers	option to prepend lines with line numbers for runjags model/data/initial value strings. This may be helpful for debugging against the output of (failed) JAGS runs. Default TRUE.
...	other options to be passed down to underlying methods where available (ignored for plot.runjags - see below).
monitor	a character vector of the names of variables to monitor. No default.
modules	external modules to be loaded into JAGS. More than 1 module can be used. Default none.
factories	factory modules to be loaded into JAGS. More than 1 factory can be used. Factories should be in the format '<facname>(<factype>)', for example: factories='mix::TemperedMix(sampler)'. Default none.
check	should the runjags object returned be checked to ensure that an external call to JAGS is able to run the model? Default TRUE.
jags	the system call or path for activating JAGS (ignored if check==FALSE). Default calls findjags() to attempt to locate JAGS on your system.

Details

The runjags class contains the full model, data, modules, factories etc required to run JAGS and is designed to encapsulate the model in a similar vein to the 'lm' or 'mer' class. Most interaction with a runjags object should be done using the print/plot/as.mcmc.list methods, but it may also be helpful to access some elements of the list directly - the names of the elements can be access using 'names(runjags.object)'. For example, this is currently the only method of extracting the full pd/popt/pd.i information.

Value

The print method for runjags objects displays a range of summary statistics for the MCMC chains (similar to that produced by summary.mcmc, but with additional details).

The plot method produces trace and density plots (note that these are pre-plotted and stored inside the runjags object, so the usual options to lattice or plot functions are not available).

The as.mcmc method combines the chains (with a warning) and returns an mcmc object, and the as.mcmc.list method extracts the mcmc.list from the runjags object (or possibly a sub-selection of variables given by vars). See also `combine.mcmc` which can be used directly on runjags objects.

The print methods for runjags model, data and initial value strings simply provide the option for printing line numbers which may be useful for debugging. The print methods for runjags.study and runjags.bginfo objects provide a basic overview of the objects.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

`combine.mcmc`,

`run.jags`,

`autorun.jags`,

`summary.mcmc`,

---

testjags *Analyse the System to Check That JAGS Is Installed*

Description

Test the users system to determine the operating system, version of R installed, and version of JAGS installed. Some information is collected from other functions such as .platform and Sys.info. Used by the run.jags function.

Usage

```
testjags(jags=findjags(), silent=FALSE)
```

Arguments

jags	the system call or path for activating JAGS. Default calls findjags() to attempt to locate JAGS on your system automatically. In unix the system call should always be 'jags', in Windows a path to the JAGS executable or the enclosing /bin or /JAGS folder is required.
silent	should on-screen feedback be suppressed? Default FALSE.

Value

A named list of values containing information about the JAGS installs found on the user's system.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

run.jags,

findjags

Examples

```
# Run the function to determine if JAGS is installed:
results <- testjags()
```

---

timestring *Calculate the Elapsed Time in Sensible Units*

Description

Function to calculate the elapsed time between 2 time periods (in seconds), or to calculate a number of seconds into a time measurement in more sensible units.

Usage

```
timestring(time1, time2=NA, units=NA, show.units=TRUE)
```

Arguments

time1	either the time index (from Sys.time()) at the start of the time period, a length of time in seconds, or an object of class 'difftime'.
time2	either the time index (from Sys.time()) at the end of the time period, or missing data if converting a single length of time. Default NA.

units	either missing, in which case a sensible time unit is chosen automatically, or one of 's', 'm', 'h', 'd', 'w', 'y' to force a specific unit. Default NA.
show.units	if TRUE, then the time is returned with units, if FALSE then just an integer is returned. Default TRUE.

Value

A time measurement, with or without units.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

Sys.time

Examples

```
# time how long it takes to complete a task:

pre.time <- Sys.time()
for (i in 1:10000000) hold <- exp(100)  # PROCESS TO TIME
post.time <- Sys.time()
timestring(pre.time, post.time)

# Convert 4687 seconds into hours:

timestring(4687, units='hours', show.units=FALSE)
```

---

xgrid.run *Remote execution of user-specified R functions on Apple Xgrid distributed computing clusters*

Description

Allows arbitrary R code to be executed on Apple Xgrid distributed computing clusters and the results returned to the R session of the user. Jobs can either be run synchronously (the process will wait for the model to complete before returning the results) or asynchronously (the process will terminate on submission of the job and results are retrieved at a later time). Access to an Xgrid cluster with R (along with all packages required by the function) installed is required. Due to the dependance on Xgrid software to perform the underlying submission and retrieval of jobs, these functions can only be used on machines running Mac OS X.

The two utility functions xgrid.jobs and xgrid.delete allow the currently running jobs to be examined and deleted from inside R.

Note Apple has discontinued Xgrid from Mac OS 10.8 onwards, so future development and testing of these functions will be extremely limited

Usage

```
xgrid.run(f=function(iteration){}, niters=1, object.list=list(),
file.list=character(0), max.threads=100, arguments=as.list(1:niters),
Rversion="", packages=list(), artfun=function() writeLines("1"),
email=NA, profiling=TRUE, cpuarch=NA, minosversion=NA,
queueforserver=FALSE, hostnode=NA, forcehost=FALSE, ramrequired=10,
jobname=NA, cleanup=TRUE, showprofiles=FALSE, Rpath='/usr/bin/R',
Rbuild='64', max.filesize="1GB",
mgridpath=system.file("xgrid", "mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), tempdir=FALSE,
keep.files=FALSE, show.output=TRUE, threads=min(niters, max.threads),
...)

xgrid.submit(f=function(iteration){}, niters=1, object.list=list(),
file.list=character(0), max.threads=100, arguments=as.list(1:niters),
Rversion="", packages=list(), artfun=function() writeLines("1"),
email=NA, profiling=TRUE, cpuarch=NA, minosversion=NA,
queueforserver=FALSE, hostnode=NA, forcehost=FALSE, ramrequired=10,
jobname=NA, Rpath='/usr/bin/R', Rbuild='64', max.filesize="1GB",
mgridpath=system.file("xgrid", "mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), show.output=TRUE,
separate.jobs=FALSE, threads=min(niters, max.threads), ...)

xgrid.results(jobinfo, wait=TRUE, partial.retrieve=!wait,
cleanup=!partial.retrieve, show.output=TRUE)

xgrid.jobs(comment=FALSE, user=FALSE, jobs=10,
mgridpath=system.file("xgrid", "mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"))

xgrid.delete(jobinfo, keep.files=FALSE)

xapply(X, FUN, method.options=list(), ...)
```

Arguments

f	the function to be iterated over on Xgrid. This must take at least 1 argument, the first of which represents the value of the 'arguments' list to be passed to the function for that iteration, which is the iteration number unless 'arguments' (or 'X' for xapply) is specified. Any other arguments to be passed to the function can be supplied as additional arguments to xgrid.run/xgrid.submit/xapply. The value(s) of interest should be returned by this function (an object of any class is

permissable). No default.

`niters` the total number of iterations over which to evaluate the function f. This can be less than the number of threads, in which case multiple iterations are evaluated serially as part of the same task. No default.

`object.list` a named list of objects that will be copied to the global environment on Xgrid and so will be visible inside the function. Alternatively, this can be a character vector of objects, that will be looked for in the global environment, rather than a named list. All other objects in the current working directory will not be visible when the function is evaluated. THIS INCLUDES LIBRARIES WHICH MUST BE RE-CALLED WITHIN THE FUNCTION BEFORE USE. In order to use functions within an R library it is therefore necessary for the required library to be installed on the Xgrid nodes on which the job will be run. If not all nodes have the required libraries installed, you can use an ART script to ensure the job is sent only to machines that do (see the example provided below), or you can use mgrid to manually request certain nodes using the '-f -h <nodename>' options. Alternatively, text files containing R code can be included in the 'file.list' argument and source()d within the function. Default blank list (no objects copied).

`file.list` a vector of filenames representing files in the current working directory that will be copied to the working directory of the executed function. This allows R code to be source()d, datasets to be loaded, and compiled code to be dynamically linked within the function, among other things. Default none.

`max.threads` the maximum number of tasks (or jobs) to split into.

`arguments` a list of values to be passed as the first argument to the function, with each element of the list specifying the value at that iteration. Default is as.list(1:niters) which passes only the iteration number to the function.

`Rversion` the required R version for worker nodes to be given tasks - may include '=' or '>=' to signify exact or minimum version requirements.

`packages` a list of R packages that must be installed on host nodes for them to be used.

`artfun` an optional user-specified R function to determine the suitability of nodes in an ART script - must either cat() 1 (indicating suitable) or 0 (indicating unsuitable) to stdout.

`email` an email address to be used to notify of job status.

`profiling` option to use ART ranking to select the most suitable host nodes preferentially.

`cpuarch` option to restrict the job to 'ppc' or 'intel' nodes.

`minosversion` option to restrict the job to nodes running a minimum Mac OS version.

`queueforserver` option to restrict the job to nodes considered to be Server machines.

`hostnode` option to prefer (or restrict to if forcehost==TRUE) running the job on the specified nodes - must be provided as a single character string with the colon character (:) separating node names.

`forcehost` option to restrict the job to only nodes specified by 'hostnode'.

`ramrequired` the minimum amount of free RAM (obtained using an approximation) for each node to be assigned a task.

jobname	the name to give the job on Xgrid (optional).
cleanup	option to remove the job from Xgrid after completion.
showprofiles	option to show the node scores based on the ART ranking used.
Rpath	the path to the R executable on the xgrid machines. If not all machines on the xgrid cluster have R (or a required package) installed then it is possible to use an ART script to ensure the job is sent to only machines that do - see the examples section for details. Default '/usr/bin/R' (this is the default install location for R).
Rbuild	the preferred binary of R to invoke. '64' results in 'Rpath64' (if it exists), '32' in 'Rpath32' (if it exists) and '' (or either of '32' or '64' if they are not found) results in Rpath. Notice that this indicates a preference, not a certainty - if the indicated build is not avalable then another will be used. Also note that specifying '64' may be ignored for PPC nodes depending on what version of R they are running (you can ensure only intel nodes are used with mgrid using sub.options='-c intel'). Default ''.
max.filesize	the maximum total size of the objects produced by the function for each thread if xgrid.method=separatejobs, or for the entire job if xgrid.method=separatetasks. This is a failsafe designed to prevent attempted transfer of huge files bringing the xgrid controller down. If the maximum size is exceeded for a thread or job then the results are erased for all iterations within that thread or job, and the job will likely have to be re-submitted. If each chain is likely to return a large amount of information, then 'separatejobs' should be used because jobs are retrieved individually which reduces the chances of overloading the Xgrid controller. The object.list is also checked to ensure it complies with the maximum size, but the file.list and any objects saved to the working directory by the function are NOT automatically cheked. Units can be provided as either "MB" or "GB". Default "1GB".
mgridpath	the path to the local mgrid script - default uses the version installed with the runjags package.
hostname	the hostname of the Xgrid server to connect to.
password	the password for the Xgrid server given by hostname.
tempdir	for xgrid.run, option to use the temporary directory as specified by the system rather than creating files in the working directory. Any files created in the temporary directory are removed when the function exits. A temporary directory cannot be used for xgrid.submit. Default TRUE when running the job synchronously.
keep.files	option to keep the folder with files needed to run the job rather than deleting it when the job is deleted from Xgrid. This may be useful for attempting to bug fix failing jobs. Default FALSE.
show.output	option to print the output of the function (obtained using cat, writeLine or print for example) at each iteration after retrieving the job(s) from xgrid. If FALSE, the output is suppressed. Default TRUE.
separate.jobs	option to submit multiple jobs to Xgrid, to help with file size constraints (see the entry for 'threads' below).
threads	the number of threads (either jobs if separate.jobs==TRUE or tasks otherwise) to generate for the job. Each thread is sent to a separate node for execution, so the

more threads there are the faster the job will finish (unless the number of threads exceeds the number of available nodes). A very large number of threads may cause problems with the Xgrid controller, hence the ability to set fewer threads than iterations. Functions that return objects of a very large size should use a large number of threads and use the xgrid.method 'separatejobs' to minimise the total size of objects returned by each xgrid job.

`...` additional arguments to be passed to the function provided by f.

`jobinfo` the output of a call to xgrid.submit.

`wait` option to wait for the Xgrid job to complete if it has not done so already.

`partial.retrieve`
for xgrid.results, option to retrieve results of partially completed jobs. By default makes cleanup FALSE. Default TRUE.

`comment` option to display any comments relevant to the Xgrid jobs running.

`user` option to display information on the user that submitted each Xgrid job.

`jobs` the number of (most recent) jobs to display information for.

`X` for xapply, a vector (atomic or list) over which to apply the function provided. Equivalent to 'arguments' for xgrid.run, with niters = length(X).

`FUN` for xapply, the function to be passed to xgrid.run as 'f'.

`method.options` for xapply, any arguments (with the exception of 'f', 'niters' and 'arguments' which are ignored) to be passed to xgrid.run.

Details

These functions allow JAGS models to be run on Xgrid distributed computing clusters from within R using the same syntax as required to run the models locally. All the functionality could be replicated by saving all necessary objects to files and using the Xgrid command line utility to submit and retrieve the job manually; these functions merely provide the convenience of not having to do this manually. Xgrid support is only available on Mac OS X machines running OS X 10.5-10.7 (Xgrid support was discontinued in Mac OS X 10.8).

The xgrid controller hostname and password can also be set as environmental variables. The command line version of R knows about environmental variables set in the .profile file, but unfortunately the GUI version does not and requires them to be set from within R using:

Sys.setenv(XGRID_CONTROLLER_HOSTNAME="<hostname>")

Sys.setenv(XGRID_CONTROLLER_PASSWORD="<password>")

(These lines could be copied into your .Rprofile file for a 'set and forget' solution)

Note that the runjags package also contains a utility shell script called 'mgrid' that enhances the capabilities of Xgrid substantially - to install this from the command line navigate to the folder given by system.file("xgrid", package="runjags") and from the terminal type 'sudo cp mgrid.sh /usr/local/bin/mgrid (or similar) to make the script visible in your search path. Help on the mgrid script can then be obtained by typing 'mgrid' (with no arguments) at the command line.

Value

For xgrid.submit, a list containing the jobname (which will be required by xgrid.results to retrieve the job) and the job ID(s) for use with the xgrid command line facilities. For xgrid.run and xgrid.results, the output of the function over all iterations is returned as a list, with each element of the list representing the results at each iteration. If the function returned an error, then the error will be held in the list as the return value at the iteration that returned the error. If the function returns an object that exceeds the 'max.filesize' when combined with the results for other iterations in that job (or greater than max.filesize/threads for multi-task jobs), the results for that thread are replaced with an error message (this is to prevent the xgrid controller crashing due to transferring large files). The xapply function returns as xgrid.run (or xgrid.submit if xgrid.options=list(submitandstop=TRUE) in which case the results can be retrieved using xgrid.results).

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

xgrid.run.jags for functions to run JAGS models on Xgrid, or run.jags to do so locally.

mclapply and parLapply in the parallel package for parallel execution of code over multiple local (or remote) cores.

Examples

```
# A basic example of synchronous running of code over 100 iterations,
# split up between 10 tasks:

## Not run:

# The function to evaluate:
f <- function(iteration){
# All objects supplied to object.list will be visible here, but
# remember to call all necessary libraries within the function

cat("Running iteration", iteration, "\n")
# Some lengthy code evaluation....

output <- rpois(10, iteration)
return(output)
}

# Run the function on xgrid for 100 iterations split between 10 machines:
results <- xgrid.run(f, niters=100, threads=10)


## End(Not run)
```

```
# A basic example of xapply to calculate the mean of a list of numbers:

## Not run:

# A list of 3 datasets from which to calculate the mean:
datasets <- list(c(1,5,6,NA), c(9,2,NA,0), c(-1,4,10,20))

# Standard lapply syntax:
results1 <- lapply(datasets, mean, na.rm=TRUE)

# Equivalent xapply syntax:
results2 <- xapply(datasets, mean,
xgrid.options=list(wait.interval='15s'), na.rm=TRUE)

# Or submit the job:
id <- xapply(datasets, mean, xgrid.options=list(submitandstop=TRUE),
na.rm=TRUE)
# And retrieve the results:
results3 <- xgrid.results(id)


## End(Not run)




## Not run:

# Subit an xgrid job just to see which packages are installed
# on a particular machine.


# A function to harvest details of R version and installed packages:
f <- function(i){

archavail <- any(dimnames(installed.packages())[[2]]=='Archs')

# To deal with older versions of R:
if(archavail){
packagesinst <- installed.packages()[,c('Version', 'Archs', 'Built')]
}else{
packagesinst <- installed.packages()[,c('Version', 'OS_type', 'Built')]
}

Rinst <- unlist(R.version[c('version.string', 'arch', 'platform')])
names(Rinst) <- c('Version', 'Archs', 'Built')
return(rbind(R=Rinst, packagesinst))

}

# Or to get more details about a particular package:
g <- function(i){
p <- library(help='bayescount')
```

```
return(p$info)
}

# Get the information back from 2 specific machines called 'newnode1'
# and 'newnode2':
results <- xgrid.run(f, niters=2, threads=2,
hostnode='newnode1:newnode2')

# See the installed packages on the two nodes:
results


## End(Not run)
```

`xgrid.run.jags` *Run a JAGS Model using an Apple Xgrid distributed computing cluster from Within R*

Description

Extends the functionality of the run.jags family of functions to use with Apple Xgrid distributed computing clusters. Jobs can either be run synchronously using xgrid.(auto)run.jags in which case the process will wait for the model to complete before returning the results, or asynchronously using xgrid.submit.jags in which case the process will terminate on submission of the job and results are retrieved at a later time using xgrid.results.jags. The latter function can also be used to check the progress of incomplete simulations without stopping or retrieving the full job. Access to an Xgrid cluster with JAGS (although not necessarily R) installed is required. Due to the dependance on Xgrid software to perform the underlying submission and retrieval of jobs, these functions can only be used on machines running Mac OS X. Further details of required environmental variables and the optional mgrid script to enable multi-task jobs can be found in the details section.

Note Apple has discontinued Xgrid from Mac OS 10.8 onwards, so future development and testing of these functions will be extremely limited

Usage

```
xgrid.run.jags(model=stop("No model supplied"), max.threads=Inf,
JAGSversion=">=2.0.0", email=NA, profiling=TRUE, cpuarch=NA,
minosversion=NA, queueforserver=FALSE, hostnode=NA, forcehost=FALSE,
ramrequired=10, jobname=NA, cleanup=TRUE, showprofiles=FALSE,
jagspath='/usr/local/bin/jags', mgridpath=system.file("xgrid",
"mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), ...)
```

```
xgrid.autorun.jags(model=stop("No model supplied"), max.threads=Inf,
JAGSversion=">=2.0.0", email=NA, profiling=TRUE, cpuarch=NA,
minosversion=NA, queueforserver=FALSE, hostnode=NA, forcehost=FALSE,
ramrequired=10, jobname=NA, cleanup=TRUE, showprofiles=FALSE,
jagspath='/usr/local/bin/jags', mgridpath=system.file("xgrid",
"mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), ...)

xgrid.extend.jags(runjags.object=stop("The output of a runjags function (with class 'runjags') must
max.threads=Inf, JAGSversion=">=2.0.0", email=NA, profiling=TRUE, cpuarch=NA,
minosversion=NA, queueforserver=FALSE, hostnode=NA, forcehost=FALSE,
ramrequired=10, jobname=NA, cleanup=TRUE, showprofiles=FALSE,
jagspath='/usr/local/bin/jags', mgridpath=system.file("xgrid",
"mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), ...)

xgrid.autoextend.jags(runjags.object=stop("The output of a runjags function (with class 'runjags')
max.threads=Inf, JAGSversion=">=2.0.0", email=NA, profiling=TRUE, cpuarch=NA,
minosversion=NA, queueforserver=FALSE, hostnode=NA, forcehost=FALSE,
ramrequired=10, jobname=NA, cleanup=TRUE, showprofiles=FALSE,
jagspath='/usr/local/bin/jags', mgridpath=system.file("xgrid",
"mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), ...)

xgrid.submit.jags(model=stop("No model supplied"), max.threads=Inf,
JAGSversion=">=2.0.0", email=NA, profiling=TRUE, cpuarch=NA,
minosversion=NA, queueforserver=FALSE, hostnode=NA, forcehost=FALSE,
ramrequired=10, jobname=NA, jagspath='/usr/local/bin/jags',
mgridpath=system.file("xgrid", "mgrid.sh", package="runjags"),
hostname=Sys.getenv("XGRID_CONTROLLER_HOSTNAME"),
password=Sys.getenv("XGRID_CONTROLLER_PASSWORD"), ...)

xgrid.results.jags(background.runjags.object=stop("An object produced by an xgrid.submit call must
wait=TRUE, cleanup=TRUE)
```

Arguments

model	a JAGS model, as would be provided to the run.jags function.
runjags.object	an object of class runjags, as would be provided to the extend.jags function.
background.runjags.object	an object of class runjags-bginfo, returned from the xgrid.submit.jags function.
max.threads	the maximum number of tasks to split the job into.
JAGSversion	the required JAGS version for worker nodes to be given tasks - may include '=' or '>=' to signify exact or minimum version requirements.

email	an email address to be used to notify of job status.
profiling	option to use ART ranking to select the most suitable host nodes preferentially.
cpuarch	option to restrict the job to 'ppc' or 'intel' nodes.
minosversion	option to restrict the job to nodes running a minimum Mac OS version.
queueforserver	option to restrict the job to nodes considered to be Server machines.
hostnode	option to prefer (or restrict to if forcehost==TRUE) running the job on the specified nodes - must be provided as a single character string with the colon character (:) separating node names.
forcehost	option to restrict the job to only nodes specified by 'hostnode'.
ramrequired	the minimum amount of free RAM (obtained using an approximation) for each node to be assigned a task.
jobname	the name to give the job on Xgrid (optional).
cleanup	option to remove the job from Xgrid after completion.
showprofiles	option to show the node scores based on the ART ranking used.
jagspath	the path to JAGS on the host nodes.
mgridpath	the path to the local mgrid script - default uses the version installed with the runjags package.
hostname	the hostname of the Xgrid server to connect to.
password	the password for the Xgrid server given by hostname.
wait	option to wait for the Xgrid job to finish if it has not already done so.
...	other options to be passed to the underlying run.jags family functions as if the model were being run locally.

Details

These functions allow JAGS models to be run on Xgrid distributed computing clusters from within R using the same syntax as required to run the models locally. All the functionality could be replicated by saving all necessary objects to files and using the Xgrid command line utility to submit and retrieve the job manually; these functions merely provide the convenience of not having to do this manually. Xgrid support is only available on Mac OS X machines running OS X 10.5-10.7 (Xgrid support was discontinued in Mac OS X 10.8).

The xgrid controller hostname and password can also be set as environmental variables. The command line version of R knows about environmental variables set in the .profile file, but unfortunately the GUI version does not and requires them to be set from within R using:

Sys.setenv(XGRID_CONTROLLER_HOSTNAME="<hostname>")

Sys.setenv(XGRID_CONTROLLER_PASSWORD="<password>")

(These lines could be copied into your .Rprofile file for a 'set and forget' solution)

Note that the runjags package also contains a utility shell script called 'mgrid' that enhances the capabilities of Xgrid substantially - to install this from the command line navigate to the folder given by system.file("xgrid", package="runjags") and from the terminal type 'sudo cp mgrid.sh /usr/local/bin/mgrid (or similar) to make the script visible in your search path. Help on the mgrid script can then be obtained by typing 'mgrid' (with no arguments) at the command line.

Value

Equivalent to that of the run.jags family of functions.

Author(s)

Matthew Denwood <matthew.denwood@glasgow.ac.uk>

See Also

run.jags, autorun.jags and runjags-class for more information on JAGS models.

xgrid.run for functions to execute user-specified functions on Xgrid.

Examples

```
# run a simple model on Xgrid using a single job:

## Not run:

# Ensure the required environmental variables are set:
Sys.setenv(XGRID_CONTROLLER_HOSTNAME="<hostname>")
Sys.setenv(XGRID_CONTROLLER_PASSWORD="<password>")

# Simulate the data
X <- 1:100
Y <- rnorm(length(X), 2*X + 10, 1)

# Model in the JAGS format
model <- "model {
for(i in 1 : N){
Y[i] ~ dnorm(true.y[i], precision);
true.y[i] <- (m * X[i]) + c;
}
m ~ dunif(-1000,1000);
c ~ dunif(-1000,1000);
precision ~ dexp(1);
}"

# Run the model synchronously using the 'simple' method:
results <- xgrid.run.jags(model=model, monitor=c("m", "c",
"precision"), data=list(N=length(X), X=X, Y=Y), n.chains=2,
plots = FALSE)

# Analyse the results:
results$summary


## End(Not run)


# Submit a job to xgrid and (later) retrieve the results.  Use an
```

```
# ART script to ensure the job is only sent to nodes with JAGS installed:

## Not run:

# Ensure the required environmental variables are set:
Sys.setenv(XGRID_CONTROLLER_HOSTNAME="<hostname>")
Sys.setenv(XGRID_CONTROLLER_PASSWORD="<password>")

# Simulate the data
X <- 1:100
Y <- rnorm(length(X), 2*X + 10, 1)

# Model in the JAGS format
model <- "model {
for(i in 1 : N){
Y[i] ~ dnorm(true.y[i], precision);
true.y[i] <- (m * X[i]) + c;
}
m ~ dunif(-1000,1000);
c ~ dunif(-1000,1000);
precision ~ dexp(1);
}"

# Run the model asynchronously:

name <- xgrid.submit.jags(model=model, monitor=c("m", "c", "precision"),
data=list(N=length(X), X=X, Y=Y), n.chains=2, plots = FALSE,
inits=list(list(.RNG.name='base::Wichmann-Hill'),
list(.RNG.name='base::Marsaglia-Multicarry')))

# Retrieve the results:
results <- xgrid.results.jags(name)


## End(Not run)
```

Index