Package 'R2jags'

May 7, 2026

Version 0.8-9

Date 2024-10-12

Title Using R to Run 'JAGS'

Author Yu-Sung Su [aut, cre] (ORCID: <https://orcid.org/0000-0001-5021-8209>),
Masanao Yajima [aut],
Gianluca Baio [ctb]

Maintainer Yu-Sung Su <suyusung@tsinghua.edu.cn>

BugReports https://github.com/suyusung/R2jags/issues/

Depends R (>= 2.14.0), rjags (>= 3-3)

Suggests testthat (>= 3.0.0)

Imports abind, coda (>= 0.13), graphics, grDevices, methods,
R2WinBUGS, parallel, stats, stringr, utils

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

Description Providing wrapper functions to implement Bayesian analysis in JAGS. Some major features include monitoring convergence of a MCMC model using Rubin and Gelman Rhat statistics, automatically running a MCMC model till it converges, and implementing parallel processing of a MCMC model for multiple chains.

License GPL (> 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2024-10-13 08:50:02 UTC

Contents

attach.jags *Attach/detach elements of 'JAGS' objects to search path*

Description

These are wraper functions for attach.bugs and detach.bugs, which attach or detach three-way-simulation array of bugs object to the search path. See attach.all for details.

Usage

```
attach.jags(x, overwrite = NA)
detach.jags()
```

Arguments

x	An rjags object.
overwrite	If TRUE, objects with identical names in the Workspace (.GlobalEnv) that are masking objects in the database to be attached will be deleted. If NA (the default) and an interactive session is running, a dialog box asks the user whether masking objects should be deleted. In non-interactive mode, behaviour is identical to overwrite=FALSE, i.e. nothing will be deleted.

Details

See attach.bugs for details

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>,

References

Sibylle Sturtz and Uwe Ligges and Andrew Gelman. (2005). "R2WinBUGS: A Package for Running WinBUGS from R." *Journal of Statistical Software* 3 (12): 1–6.

Examples

```
# See the example in ?jags for the usage.
```

autojags *Function for auto-updating 'JAGS' until the model converges*

Description

The autojags takes a rjags object as input. autojags will update the model until it converges.

Usage

```
## S3 method for class 'rjags'
update(object, n.iter=1000, n.thin=1,
                      refresh=n.iter/50, progress.bar = "text", ...)
autojags(object, n.iter=1000, n.thin=1, Rhat=1.1, n.update=2,
      refresh=n.iter/50, progress.bar = "text", ...)
```

Arguments

object	an object of rjags class.
n.iter	number of total iterations per chain, default=1000
n.thin	thinning rate. Must be a positive integer, default=1
...	further arguments pass to or from other methods.
Rhat	converegence criterion, default=1.1.
n.update	the max number of updates, default=2.
refresh	refresh frequency for progress bar, default is n.iter/50
progress.bar	type of progress bar. Possible values are "text", "gui", and "none". Type "text" is displayed on the R console. Type "gui" is a graphical progress bar in a new window. The progress bar is suppressed if progress.bar is "none"

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>

References

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. (2003): *Bayesian Data Analysis*, 2nd edition, CRC Press.

Examples

```
# see ?jags for an example.
```

jags *Run 'JAGS' from R*

Description

The jags function takes data and starting values as input. It automatically writes a jags script, calls the model, and saves the simulations for easy access in R.

Usage

```
jags(data, inits, parameters.to.save, model.file="model.bug",
  n.chains=3, n.iter=2000, n.burnin=floor(n.iter/2),
  n.thin=max(1, floor((n.iter - n.burnin) / 1000)),
  DIC=TRUE, pD = FALSE, n.iter.pd = NULL, n.adapt = 100,
  working.directory=NULL, jags.seed = 123,
  refresh = n.iter/50, progress.bar = "text", digits=5,
  RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry",
              "Super-Duper", "Mersenne-Twister"),
  jags.module = c("glm","dic"), quiet = FALSE,
  checkMissing = FALSE
  )

jags.parallel(data, inits, parameters.to.save, model.file = "model.bug",
              n.chains = 2, n.iter = 2000, n.burnin = floor(n.iter/2),
              n.thin = max(1, floor((n.iter - n.burnin)/1000)),
              n.cluster= n.chains, DIC = TRUE,
              working.directory = NULL, jags.seed = 123, digits=5,
              RNGname = c("Wichmann-Hill", "Marsaglia-Multicarry",
               "Super-Duper", "Mersenne-Twister"),
              jags.module = c("glm","dic"),
              export_obj_names=NULL,
              envir = .GlobalEnv
              )

jags2(data, inits, parameters.to.save, model.file="model.bug",
  n.chains=3, n.iter=2000, n.burnin=floor(n.iter/2),
  n.thin=max(1, floor((n.iter - n.burnin) / 1000)),
  DIC=TRUE, jags.path="",
  working.directory=NULL, clearWD=TRUE,
  refresh = n.iter/50)
```

Arguments

data	(1) a vector or list of the names of the data objects used by the model, (2) a (named) list of the data objects themselves, or (3) the name of a "dump" format file containing the data objects, which must end in ".txt", see example below for details.

inits — a list with n.chains elements; each element of the list is itself a list of starting values for the BUGS model, *or* a function creating (possibly random) initial values. If inits is NULL, JAGS will generate initial values for parameters.

parameters.to.save — character vector of the names of the parameters to save which should be monitored.

model.file — file containing the model written in BUGS code. Alternatively, as in **R2WinBUGS**, model.file can be an R function that contains a BUGS model that is written to a temporary model file (see tempfile) using write.model

n.chains — number of Markov chains (default: 3)

n.iter — number of total iterations per chain (including burn in; default: 2000)

n.burnin — length of burn in, i.e. number of iterations to discard at the beginning. Default is n.iter/2, that is, discarding the first half of the simulations. If n.burnin is 0, jags() will run 100 iterations for adaption.

n.cluster — number of clusters to use to run parallel chains. Default equals n.chains.

n.thin — thinning rate. Must be a positive integer. Set n.thin > 1 to save memory and computation time if n.iter is large. Default is max(1, floor(n.chains * (n.iter-n.burnin) / 1000)) which will only thin if there are at least 2000 simulations.

DIC — logical; if TRUE (default), compute deviance, pD, and DIC. The rule pD=var(deviance) / 2 is used.

pD — logical; if TRUE and DIC is also TRUE, then adds the computation of 'pD', using 'rjags::dic.samples()'. Defaults to FALSE.

n.iter.pd — number of iterations to feed 'rjags::dic.samples()' to compute 'pD'. Defaults at 1000.

n.adapt — number of iterations for which to run the adaptation, when creating the model object. Defaults at 100.

working.directory — sets working directory during execution of this function; This should be the directory where model file is.

jags.seed — random seed for JAGS, default is 123. This function is used for jags.parallell() and does not work for jags(). Use set.seed() instead if you want to produce identical result with jags()

.

jags.path — directory that contains the JAGS executable. The default is "".

clearWD — indicating whether the files 'data.txt', 'inits[1:n.chains].txt', 'codaIndex.txt', 'jagsscript.txt', and 'CODAchain[1:nchains].txt' should be removed after jags has finished, default=TRUE.

refresh — refresh frequency for progress bar, default is n.iter/50

progress.bar — type of progress bar. Possible values are "text", "gui", and "none". Type "text" is displayed on the R console. Type "gui" is a graphical progress bar in a new window. The progress bar is suppressed if progress.bar is "none"

digits	as in write.model in the **R2WinBUGS** package: number of significant digits used for BUGS input, see formatC. Only used if specifying a BUGS model as an R function.
RNGname	the name for random number generator used in JAGS. There are four RNGS supplied by the base moduale in JAGS: Wichmann-Hill, Marsaglia-Multicarry, Super-Duper, Mersenne-Twister
jags.module	the vector of jags modules to be loaded. Default are "glm" and "dic". Input NULL if you don't want to load any jags module.
export_obj_names	character vector of objects to export to the clusters.
envir	default is .GlobalEnv
quiet	Logical, whether to suppress stdout in jags.model().
checkMissing	Default: FALSE. When TRUE, checks for missing data in categorical parameters and returns a sim.list with NA values if detected. It's recommended to supply jags() with complete data.

Details

To run:

1. Write a JAGS model in an ASCII file.
2. Go into R.
3. Prepare the inputs for the jags function and run it (see Example section).
4. The model will now run in JAGS. It might take awhile. You will see things happening in the R console.

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>, Masanao Yajima <yajima@bu.edu>, Gianluca Baio <g.baio@ucl.ac.uk>

References

Plummer, Martyn (2003) "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." https://www.r-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf.

Gelman, A., Carlin, J. B., Stern, H.S., Rubin, D.B. (2003) *Bayesian Data Analysis*, 2nd edition, CRC Press.

Sibylle Sturtz and Uwe Ligges and Andrew Gelman. (2005). "R2WinBUGS: A Package for Running WinBUGS from R." *Journal of Statistical Software* 3 (12): 1–6.

Examples

```
# An example model file is given in:
model.file <- system.file(package="R2jags", "model", "schools.txt")
# Let's take a look:
file.show(model.file)
# you can also write BUGS model as a R function, see below:
```

```
#=================#
# initialization  #
#=================#

  # data
  J <- 8.0
  y <- c(28.4,7.9,-2.8,6.8,-0.6,0.6,18.0,12.2)
  sd <- c(14.9,10.2,16.3,11.0,9.4,11.4,10.4,17.6)


  jags.data <- list("y","sd","J")
  jags.params <- c("mu","sigma","theta")
  jags.inits <- function(){
    list("mu"=rnorm(1),"sigma"=runif(1),"theta"=rnorm(J))
  }

  ## You can input data in 4 ways
  ## 1) data as list of character
  jagsfit <- jags(data=list("y","sd","J"), inits=jags.inits, jags.params,
                n.iter=10, model.file=model.file)

  ## 2) data as character vector of names
  jagsfit <- jags(data=c("y","sd","J"), inits=jags.inits, jags.params,
                n.iter=10, model.file=model.file)

  ## 3) data as named list
  jagsfit <- jags(data=list(y=y,sd=sd,J=J), inits=jags.inits, jags.params,
                n.iter=10, model.file=model.file)

  ## 4) data as a file
  fn <- "tmpbugsdata.txt"
  dump(c("y","sd","J"), file=fn)
  jagsfit <- jags(data=fn, inits=jags.inits, jags.params,
                  n.iter=10, model.file=model.file)
  unlink("tmpbugsdata.txt")

  ## You can write bugs model in R as a function

  schoolsmodel <- function() {
    for (j in 1:J){                      # J=8, the number of schools
      y[j] ~ dnorm (theta[j], tau.y[j]) # data model:  the likelihood
      tau.y[j] <- pow(sd[j], -2)         # tau = 1/sigma^2
    }
    for (j in 1:J){
      theta[j] ~ dnorm (mu, tau)         # hierarchical model for theta
    }
    tau <- pow(sigma, -2)                # tau = 1/sigma^2
    mu ~ dnorm (0.0, 1.0E-6)             # noninformative prior on mu
    sigma ~ dunif (0, 1000)              # noninformative prior on sigma
  }

  jagsfit <- jags(data=jags.data, inits=jags.inits, jags.params,
```

```
                n.iter=10, model.file=schoolsmodel)


#================================#
# RUN jags and postprocessing   #
#================================#
  jagsfit <- jags(data=jags.data, inits=jags.inits, jags.params,
    n.iter=5000, model.file=model.file)

  # Can also compute the DIC using pD (=Dbar-Dhat), via dic.samples(), which
  # is a closer approximation to the original formulation of Spiegelhalter et
  # al (2002), instead of pV (=var(deviance)/2), which is the default in JAGS
   jagsfit.pD <- jags(data=jags.data, inits=jags.inits, jags.params,
     n.iter=5000, model.file=model.file, pD=TRUE)

  # Run jags parallely, no progress bar. R may be frozen for a while,
  # Be patient. Currenlty update afterward does not run parallelly
  #
   jagsfit.p <- jags.parallel(data=jags.data, inits=jags.inits, jags.params,
     n.iter=5000, model.file=model.file)

  # display the output
  print(jagsfit)
  plot(jagsfit)

  # traceplot
  traceplot(jagsfit.p)
  traceplot(jagsfit)

  # or to use some plots in coda
  # use as.mcmmc to convert rjags object into mcmc.list
  jagsfit.mcmc <- as.mcmc(jagsfit.p)
  jagsfit.mcmc <- as.mcmc(jagsfit)
  ## now we can use the plotting methods from coda
  #require(lattice)
  #xyplot(jagsfit.mcmc)
  #densityplot(jagsfit.mcmc)

  # if the model does not converge, update it!
  jagsfit.upd <- update(jagsfit, n.iter=100)
  print(jagsfit.upd)
  print(jagsfit.upd, intervals=c(0.025, 0.5, 0.975))
  plot(jagsfit.upd)

  # before update parallel jags object, do recompile it
  recompile(jagsfit.p)
  jagsfit.upd <- update(jagsfit.p, n.iter=100)



  # or auto update it until it converges! see ?autojags for details
  # recompile(jagsfit.p)
  jagsfit.upd <- autojags(jagsfit.p)
```

```
  jagsfit.upd <- autojags(jagsfit)

  # to get DIC or specify DIC=TRUE in jags() or do the following#
  dic.samples(jagsfit.upd$model, n.iter=1000, type="pD")

  # attach jags object into search path see "attach.bugs" for details
  attach.jags(jagsfit.upd)

  # this will show a 3-way array of the bugs.sim object, for example:
  mu

  # detach jags object into search path see "attach.bugs" for details
  detach.jags()

  # to pick up the last save session
  # for example, load("RWorkspace.Rdata")
  recompile(jagsfit)
  jagsfit.upd <- update(jagsfit, n.iter=100)

  recompile(jagsfit.p)
  jagsfit.upd <- update(jagsfit, n.iter=100)

#=============#
# using jags2 #
#=============#
  ## jags can be run and produces coda files, but cannot be updated once it's done
  ## You may need to edit "jags.path" to make this work,
  ## also you need a write access in the working directory:
  ## e.g. setwd("d:/")

  ## NOT RUN HERE
  ## Not run:
    jagsfit <- jags2(data=jags.data, inits=jags.inits, jags.params,
      n.iter=5000, model.file=model.file)
    print(jagsfit)
    plot(jagsfit)
    # or to use some plots in coda
    # use as.mcmmc to convert rjags object into mcmc.list
    jagsfit.mcmc <- as.mcmc.list(jagsfit)
    traceplot(jagsfit.mcmc)
    #require(lattice)
    #xyplot(jagsfit.mcmc)
    #densityplot(jagsfit.mcmc)

## End(Not run)
```

---

jags2bugs *Read jags output files in CODA format*

Description

This function reads Markov Chain Monte Carlo output in the CODA format produced by **jags** and returns an object of class `mcmc.list` for further output analysis using the **coda** package.

Usage

```
jags2bugs(path=getwd(), parameters.to.save,
  n.chains=3, n.iter=2000, n.burnin=1000, n.thin=2,
  DIC=TRUE)
```

Arguments

`path`	sets working directory during execution of this function; This should be the directory where CODA files are.
`parameters.to.save`	character vector of the names of the parameters to save which should be monitored.
`n.chains`	number of Markov chains (default: 3)
`n.iter`	number of total iterations per chain (including burn in; default: 2000)
`n.burnin`	length of burn in, i.e. number of iterations to discard at the beginning. Default is `n.iter/2`, that is, discarding the first half of the simulations.
`n.thin`	thinning rate, default is 2
`DIC`	logical; if `TRUE` (default), compute deviance, pD, and DIC. The rule `pD=var(deviance) / 2` is used.

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>, Masanao Yajima <yajima@stat.columbia.edu>

`recompile` *Function for recompiling rjags object*

Description

The `recompile` takes a `rjags` object as input. `recompile` will re-compile the previous saved `rjags` object.

Usage

```
recompile(object, n.iter, refresh, progress.bar)
## S3 method for class 'rjags'
recompile(object, n.iter=100, refresh=n.iter/50,
    progress.bar = "text")
```

Arguments

object	an object of rjags class.
n.iter	number of iteration for adapting, default is 100
refresh	refresh frequency for progress bar, default is n.iter/50
progress.bar	type of progress bar. Possible values are "text", "gui", and "none". Type "text" is displayed on the R console. Type "gui" is a graphical progress bar in a new window. The progress bar is suppressed if progress.bar is "none"

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>

Examples

```
# see ?jags for an example.
```

---

traceplot *Trace plot of bugs object*

Description

Displays a plot of iterations *vs.* sampled values for each variable in the chain, with a separate plot per variable.

Usage

```
traceplot(x, ...)
## S4 method for signature 'rjags'
traceplot(x, mfrow = c(1, 1), varname = NULL,
  match.head = TRUE, ask = TRUE,
  col = rainbow( x$n.chains ),
  lty = 1, lwd = 1, ...)
```

Arguments

x	A bugs object
mfrow	graphical parameter (see par)
varname	vector of variable names to plot
match.head	matches the variable names by the beginning of the variable names in bugs object
ask	logical; if TRUE, the user is *ask*ed before each plot, see par(ask=.).
col	graphical parameter (see par)
lty	graphical parameter (see par)
lwd	graphical parameter (see par)
...	further graphical parameters

Author(s)

Masanao Yajima <yajima@stat.columbia.edu>.

See Also

densplot, plot.mcmc, traceplot

Index