Package: jagsUI (via r-universe)

October 15, 2026

Version 1.6.3

Date 2026-01-09

Title A Wrapper Around 'rjags' to Streamline 'JAGS' Analyses

Depends R (>= 2.14.0),

Imports coda (>= 0.13), graphics, grDevices, parallel, rjags (>= 4-17), stats, utils

Suggests knitr, markdown, tinytest

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

Description A set of wrappers around 'rjags' functions to run Bayesian analyses in 'JAGS' (specifically, via 'libjags'). A single function call can control adaptive, burn-in, and sampling MCMC phases, with MCMC chains run in sequence or in parallel. Posterior distributions are automatically summarized (with the ability to exclude some monitored nodes if desired) and functions are available to generate figures based on the posteriors (e.g., predictive check plots, traceplots). Function inputs, argument syntax, and output format are nearly identical to the 'R2WinBUGS'/'R2OpenBUGS' packages to allow easy switching between MCMC samplers.

License GPL-3

URL https://kenkellner.com/jagsUI/

BugReports https://github.com/kenkellner/jagsUI/issues

NeedsCompilation no

VignetteBuilder knitr

Author Ken Kellner [cre, aut], Mike Meredith [ctb]

Maintainer Ken Kellner <contact@kenkellner.com>

Config/pak/sysreqs jags

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

Date/Publication 2026-01-10 08:20:02 UTC

RemoteUrl https://github.com/cran/jagsUI

RemoteRef HEAD

RemoteSha 4191063bca9a8a51963133c98ac0d62ccf4391c1

Contents

autojags 2
densityplot 4
jags 5
jags.basic 9
jags.View 11
pp.check 11
traceplot 13
update 14
whiskerplot 15

Index **18**

`autojags` *Automatically run jagsUI analyses to convergence*

Description

The `autojags` function runs repeated updates of `jagsUI` models, until a specified convergence level (based on the statistic Rhat) or a maximum number of iterations is reached.

Usage

```
autojags(data, inits, parameters.to.save, model.file,
  n.chains, n.adapt=NULL, iter.increment=1000, n.burnin=0, n.thin=1,
  save.all.iter=FALSE, modules=c('glm'), factories=NULL,
  parallel=FALSE, n.cores=NULL, DIC=TRUE,
  store.data=FALSE, codaOnly=FALSE,seed=NULL,
  bugs.format=FALSE, Rhat.limit=1.1, max.iter=100000, verbose=TRUE)
```

Arguments

`data`	A named list of the data objects required by the model, or a character vector containing the names of the data objects required by the model. Use of a character vector will be deprecated in the next version - switch to using named lists.
`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 in the model which should be monitored.
`model.file`	Path to file containing the model written in `BUGS` code

n.chains	Number of Markov chains to run.
n.adapt	Number of iterations to run in the JAGS adaptive phase. The default is NULL, which will result in the function running groups of 100 adaptation iterations (to a max of 10,000) until JAGS reports adaptation is sufficient. If you set n.adapt manually, 1000 is the recommended minimum value.
iter.increment	Number of iterations per model auto-update. Set to larger values when you suspect the model will take a long time to converge.
n.burnin	Number of iterations at the beginning of the chain to discard (i.e., the burn-in). Does not include the adaptive phase iterations.
n.thin	Thinning rate. Must be a positive integer.
save.all.iter	Option to combine MCMC samples from all iterative updates into final posterior (by default only the final iteration is included in the posterior).
modules	List of JAGS modules to load before analysis. By default only module 'glm' is loaded (in addition to 'basemod' and 'bugs'). To force no additional modules to load, set modules=NULL.
factories	Optional character vector of factories to enable or disable, in the format <factory> <type> <setting>. For example, to turn TemperedMix on you would provide 'mix::TemperedMix sampler TRUE' (note spaces between parts). Make sure you have the corresponding modules loaded as well.
parallel	If TRUE, run MCMC chains in parallel on multiple CPU cores
n.cores	If parallel=TRUE, specify the number of CPU cores used. Defaults to total available cores or the number of chains, whichever is smaller.
DIC	Option to report DIC and the estimated number of parameters (pD). Defaults to TRUE.
store.data	Option to store the input dataset and initial values in the output object for future use. Defaults to FALSE.
codaOnly	Optional character vector of parameter names for which you do NOT want to calculate detailed statistics. This may be helpful when you have many output parameters (e.g., predicted values) and you want to save time. For these parameters, only the mean value will be calculated but the mcmc output will still be found in $sims.list and $samples.
seed	Option to set a custom seed to initialize JAGS chains, for reproducibility. Should be an integer. This argument will be deprecated in the next version, but you can always set the outside the function yourself.
bugs.format	Option to print JAGS output in classic R2WinBUGS format. Default is FALSE.
Rhat.limit	Set the desired cutoff point for convergence; when all Rhat values are less than this value the model assumes convergence has been reached and will stop auto-updating.
max.iter	Maximum number of total iterations allowed via auto-update (including burn-in).
verbose	If set to FALSE, all text output in the console will be suppressed as the function runs (including most warnings).

Details

Usage and output is otherwise identical to the jags function.

Author(s)

Ken Kellner <contact@kenkellner.com>.

---

densityplot *Density plots of JAGS output*

Description

Displays a series of density plots for posteriors of monitored parameters in a JAGS analysis.

Usage

```
densityplot(x, parameters=NULL, layout=NULL, ask=NULL)
```

Arguments

x	A jagsUI object
parameters	A vector of names (as characters) of parameters to plot. Parameter names must match parameters included in the model. Calling non-scalar parameters without subsetting (e.g. alpha) will plot all values of alpha. If parameters=NULL, all parameters will be plotted.
layout	A length 2 vector with the number of rows and columns to display in the plot. The default is 3 x 3, or smaller if there are fewer parameters to plot.
ask	If TRUE, ask user for confirmation before generating each new plot; the default is to ask when output is going to the screen, not when it is going to a file.

Author(s)

Ken Kellner <contact@kenkellner.com>.

jags *Call JAGS from R*

Description

The jags function is a basic user interface for running JAGS analyses via package rjags inspired by similar packages like R2WinBUGS, R2OpenBUGS, and R2jags. The user provides a model file, data, initial values (optional), and parameters to save. The function compiles the information and sends it to JAGS, then consolidates and summarizes the MCMC output in an object of class jagsUI.

Usage

```
jags(data, inits, parameters.to.save, model.file,
  n.chains, n.adapt=NULL, n.iter, n.burnin=0, n.thin=1,
  modules=c('glm'), factories=NULL, parallel=FALSE,
  n.cores=NULL, DIC=TRUE, store.data=FALSE,
  codaOnly=FALSE,seed=NULL, bugs.format=FALSE, verbose=TRUE)
```

Arguments

data	A named list of the data objects required by the model, or a character vector containing the names of the data objects required by the model. Use of a character vector will be deprecated in the next version - switch to using named lists.
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 in the model which should be monitored.
model.file	Path to file containing the model written in BUGS code
n.chains	Number of Markov chains to run.
n.adapt	Number of iterations to run in the JAGS adaptive phase. The default is NULL, which will result in the function running groups of 100 adaptation iterations (to a max of 10,000) until JAGS reports adaptation is sufficient. If you set n.adapt manually, 1000 is the recommended minimum value.
n.iter	Total number of iterations per chain (including burn-in).
n.burnin	Number of iterations at the beginning of the chain to discard (i.e., the burn-in). Does not include the adaptive phase iterations.
n.thin	Thinning rate. Must be a positive integer.
modules	List of JAGS modules to load before analysis. By default only module 'glm' is loaded (in addition to 'basemod' and 'bugs'). To force no additional modules to load, set modules=NULL.

factories	Optional character vector of factories to enable or disable, in the format <factory> <type> <setting>. For example, to turn TemperedMix on you would provide 'mix::TemperedMix sampler TRUE' (note spaces between parts). Make sure you have the corresponding modules loaded as well.
parallel	If TRUE, run MCMC chains in parallel on multiple CPU cores
n.cores	If parallel=TRUE, specify the number of CPU cores used. Defaults to total available cores or the number of chains, whichever is smaller.
DIC	Option to report DIC and the estimated number of parameters (pD). Defaults to TRUE.
store.data	Option to store the input dataset and initial values in the output object for future use. Defaults to FALSE.
codaOnly	Optional character vector of parameter names for which you do NOT want to calculate detailed statistics. This may be helpful when you have many output parameters (e.g., predicted values) and you want to save time. For these parameters, only the mean value will be calculated but the mcmc output will still be found in $sims.list and $samples.
seed	Option to set a custom seed to initialize JAGS chains, for reproducibility. Should be an integer. This argument will be deprecated in the next version, but you can always set the outside the function yourself.
bugs.format	Option to print JAGS output in classic R2WinBUGS format. Default is FALSE.
verbose	If set to FALSE, all text output in the console will be suppressed as the function runs (including most warnings).

Details

Basic analysis steps:

1. Collect and package data
2. Write a model file in BUGS language
3. Set initial values
4. Specify parameters to monitor
5. Set MCMC variables and run analysis
6. Optionally, generate more posterior samples using the update method.

See example below.

Value

An object of class jagsUI. Notable elements in the output object include:

sims.list	A list of values sampled from the posterior distributions of each monitored parameter.
summary	A summary of various statistics calculated based on model output, in matrix form.
samples	The original output object from the rjags package, as class mcmc.list.
model	The rjags model object; this will contain multiple elements if parallel=TRUE.

Author(s)

Ken Kellner <contact@kenkellner.com>.

Examples

```
#Analyze Longley economic data in JAGS

#Number employed as a function of GNP

#####################################
##   1. Collect and Package Data    ##
#####################################

#Load data (built into R)

data(longley)
head(longley)

#Separate data objects

gnp <- longley$GNP
employed <- longley$Employed
n <- length(employed)

#Input data objects must be numeric, and must be
#scalars, vectors, matrices, or arrays.

#Package together
data <- list(gnp=gnp,employed=employed,n=n)

#####################################
##      2. Write model file        ##
#####################################

#Write a model in the BUGS language

#Generate model file directly in R
#(could also read in existing model file)

#Identify filepath of model file
modfile <- tempfile()

#Write model to file
writeLines("
model{

  #Likelihood
  for (i in 1:n){

    employed[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha + beta*gnp[i]
```

```
  }

  #Priors
  alpha ~ dnorm(0, 0.00001)
  beta ~ dnorm(0, 0.00001)
  sigma ~ dunif(0,1000)
  tau <- pow(sigma,-2)

}
", con=modfile)

#####################################
##    3. Initialize Parameters     ##
#####################################

#Best to generate initial values using function

inits <- function(){
  list(alpha=rnorm(1,0,1),beta=rnorm(1,0,1),sigma=runif(1,0,3))
}

#In many cases, JAGS can pick initial values automatically;
#you can leave argument inits=NULL to allow this.

#####################################
##  4. Set parameters to monitor   ##
#####################################

#Choose parameters you want to save output for
#Only parameters in this list will appear in output object
#(deviance is added automatically if DIC=TRUE)

#List must be specified as a character vector

params <- c('alpha','beta','sigma')

#####################################
##        5. Run Analysis          ##
#####################################

#Call jags function; specify number of chains, number of adaptive iterations,
#the length of the burn-in period, total iterations, and the thin rate.

out <- jags(data = data,
            inits = inits,
            parameters.to.save = params,
            model.file = modfile,
            n.chains = 3,
            n.adapt = 100,
            n.iter = 1000,
            n.burnin = 500,
            n.thin = 2)
```

```
#Arguments will be passed to JAGS; you will see progress bars
#and other information

#Examine output summary

out

#Look at output object elements
names(out)

#Plot traces and posterior densities
plot(out)

#Plot traces
traceplot(out)

#Update model another 1000 iterations
out <- update(out,n.iter = 1000)
```

---

jags.basic *Simplified function to call JAGS from R*

Description

The jags.basic function is a simplified version of the jags function which returns only the mcmc.list-class output from rjags rather than a more complex summary (it will also optionally return the model, in which case the output object will be class jagsUIbasic). This minimal function may be useful when the input dataset or output parameter set are very large and memory intensive.

Usage

```
jags.basic(data, inits, parameters.to.save, model.file,
  n.chains, n.adapt=NULL, n.iter, n.burnin=0, n.thin=1,
  modules=c('glm'), factories=NULL, parallel=FALSE, n.cores=NULL, DIC=TRUE,
  seed=NULL, save.model=FALSE, verbose=TRUE)
```

Arguments

data	A named list of the data objects required by the model, or a character vector containing the names of the data objects required by the model. Use of a character vector will be deprecated in the next version - switch to using named lists.
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 in the model which should be monitored.

model.file	Path to file containing the model written in BUGS code
n.chains	Number of Markov chains to run.
n.adapt	Number of iterations to run in the JAGS adaptive phase. The default is NULL, which will result in the function running groups of 100 adaptation iterations (to a max of 10,000) until JAGS reports adaptation is sufficient. If you set n.adapt manually, 1000 is the recommended minimum value.
n.iter	Total number of iterations per chain (including burn-in).
n.burnin	Number of iterations at the beginning of the chain to discard (i.e., the burn-in). Does not include the adaptive phase iterations.
n.thin	Thinning rate. Must be a positive integer.
modules	List of JAGS modules to load before analysis. By default only module 'glm' is loaded (in addition to 'basemod' and 'bugs'). To force no additional modules to load, set modules=NULL.
factories	Optional character vector of factories to enable or disable, in the format <factory> <type> <setting>. For example, to turn TemperedMix on you would provide 'mix::TemperedMix sampler TRUE' (note spaces between parts). Make sure you have the corresponding modules loaded as well.
parallel	If TRUE, run MCMC chains in parallel on multiple CPU cores
n.cores	If parallel=TRUE, specify the number of CPU cores used. Defaults to total available cores or the number of chains, whichever is smaller.
DIC	Option to report deviance values. Defaults to TRUE.
seed	Option to set a custom seed to initialize JAGS chains, for reproducibility. Should be an integer. This argument will be deprecated in the next version, but you can always set the outside the function yourself.
save.model	Returns the JAGS model as part of the output object to allow updating the model later. If TRUE, the output object will instead be a list of class jagsUIbasic. Default is false.
verbose	If set to FALSE, all text output in the console will be suppressed as the function runs (including most warnings).

Details

See documentation for jags function for analysis details. The update method will only work if save.model=TRUE.

Value

An object of class mcmc.list, if save.model=FALSE; if save.model=TRUE, a 2-element list of class jagsUIbasic containing the mcmc samples and the model.

Author(s)

Ken Kellner <contact@kenkellner.com>.

jags.View — *View a jagsUI output object in a separate window*

Description

Show an R object in a separate, spreadsheet-style window via a call to `View`.

Usage

```
jags.View(x, title, digits=3)
```

Arguments

`x`	A jagsUI object
`title`	Specify a title for the window.
`digits`	Number of digits to display after the decimal.

Author(s)

Ken Kellner <contact@kenkellner.com> and Mike Meredith.

pp.check — *Posterior Predictive Checks for Bayesian Analyses fit in JAGS*

Description

A simple interface for generating a posterior predictive check plot for a JAGS analysis fit using jagsUI, based on the posterior distributions of discrepency metrics specified by the user and calculated and returned by JAGS (for example, sums of residuals). The user supplies the name of the discrepancy metric calculated for the real data in the argument `observed`, and the corresponding discrepancy for data simulated by the model in argument `simulated`. The posterior distributions of the two parameters will be plotted in X-Y space and a Bayesian p-value calculated.

Usage

```
pp.check(x, observed, simulated, xlab='Observed data', ylab='Simulated data',
                  main='Posterior Predictive Check', ...)
```

Arguments

x	A jagsUI object generated using the jags function
observed	The name of the parameter (as a string) representing the fit of the observed data (e.g. residuals)
simulated	The name of the corresponding parameter (as a string) representing the fit of the new simulated data
xlab	Customize x-axis label
ylab	Customize y-axis label
main	Customize plot title
...	Additional arguments passed to plot.default

Author(s)

Ken Kellner <contact@kenkellner.com>.

Examples

```
#Analyze Longley economic data in JAGS
#Number employed as a function of GNP
#See ?jags for a more detailed example

#Get data
data(longley)
gnp <- longley$GNP
employed <- longley$Employed
n <- length(employed)
data <- list(gnp=gnp,employed=employed,n=n)

#Identify filepath of model file
modfile <- tempfile()

#Write model
#Note calculation of discrepancy stats fit and fit.new
#(sums of residuals)
writeLines("
model{

  #Likelihood
  for (i in 1:n){

    employed[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha + beta*gnp[i]

    res[i] <- employed[i] - mu[i]
    emp.new[i] ~ dnorm(mu[i], tau)
    res.new[i] <- emp.new[i] - mu[i]

  }
```

```
  #Priors
  alpha ~ dnorm(0, 0.00001)
  beta ~ dnorm(0, 0.00001)
  sigma ~ dunif(0,1000)
  tau <- pow(sigma,-2)

  #Derived parameters
  fit <- sum(res[])
  fit.new <- sum(res.new[])

}
", con=modfile)

#Set parameters to monitor
params <- c('alpha','beta','sigma','fit','fit.new')

#Run analysis

out <- jags(data = data,
            inits = NULL,
            parameters.to.save = params,
            model.file = modfile,
            n.chains = 3,
            n.adapt = 100,
            n.iter = 1000,
            n.burnin = 500,
            n.thin = 2)

#Examine output summary

out

#Posterior predictive check plot

pp.check(out, observed = 'fit', simulated = 'fit.new')
```

`traceplot` *Traceplots of JAGS output*

Description

Displays a series of MCMC iteration plots for monitored parameter in a JAGS analysis, along with the calculated Rhat value.

Usage

```
traceplot(x, parameters=NULL, Rhat_min=NULL, layout=NULL, ask=NULL)
```

Arguments

x	A jagsUI object
parameters	A vector of names (as characters) of parameters to plot. Parameter names must match parameters included in the model. Calling non-scalar parameters without subsetting (e.g. alpha) will plot all values of alpha. If parameters=NULL, all parameters will be plotted.
Rhat_min	If provided, only plot parameters with Rhat values that exceed the provided value. A good min value to start with is 1.05.
layout	A length 2 vector with the number of rows and columns to display in the plot. The default is 3 x 3, or smaller if there are fewer parameters to plot.
ask	If TRUE, ask user for confirmation before generating each new plot; the default is to ask when output is going to the screen, not when it is going to a file.

Author(s)

Ken Kellner <contact@kenkellner.com>.

---

update *Update a JAGS model*

Description

This function updates a JAGS model created by created by function jags in package jagsUI for a specified number of iterations.

Usage

```
  ## S3 method for class 'jagsUI'
update(object, parameters.to.save=NULL, n.adapt=NULL,
  n.iter, n.thin=NULL, modules=c('glm'), factories=NULL, DIC=NULL,
  codaOnly=FALSE, verbose=TRUE, ...)
```

Arguments

object	A jagsUI or jagsUIbasic-class object to update.
parameters.to.save	Character vector of the names of the parameters in the model which should be monitored. Defaults to the saved parameter set from the original model run.
n.adapt	Number of iterations to run in the JAGS adaptive phase. The default is NULL, which will result in the function running groups of 100 adaptation iterations (to a max of 10,000) until JAGS reports adaptation is sufficient. If you set n.adapt manually, 1000 is the recommended minimum value.
n.iter	Number of iterations to update for each chain.
n.thin	Thinning rate. Must be a positive integer. Defaults to the thinning rate of the original model run.

modules	List of JAGS modules to load before analysis. By default only module 'glm' is loaded (in addition to 'basemod' and 'bugs'). To force no additional modules to load, set modules=NULL.
factories	Optional character vector of factories to enable or disable, in the format <factory> <type> <setting>. For example, to turn TemperedMix on you would provide 'mix::TemperedMix sampler TRUE' (note spaces between parts). Make sure you have the corresponding modules loaded as well.
DIC	Option to report DIC and the estimated number of parameters (pD). Defaults to the same setting as the original model to updated.
codaOnly	Optional character vector of parameter names for which you do NOT want to calculate detailed statistics. This may be helpful when you have many output parameters (e.g., predicted values) and you want to save time. For these parameters, only the mean value will be calculated but the mcmc output will still be found in $sims.list and $samples.
verbose	If set to FALSE, all text output in the console will be suppressed as the function runs (including most warnings).
...	Further arguments pass to or from other methods.

Author(s)

Ken Kellner <contact@kenkellner.com>.

---

whiskerplot *Whisker plots of parameter posterior distributions*

Description

Displays whisker plots for specified parameters on the same plot, with a point at the mean value for the posterior distribution and whiskers extending to the specified quantiles of the distribution.

Usage

```
whiskerplot(x, parameters, quantiles=c(0.025,0.975), zeroline=TRUE, ...)
```

Arguments

x	A jagsUI object
parameters	A vector of names (as characters) of parameters to include in the plot. Parameter names must match parameters included in the model. Calling non-scalar parameters without subsetting (e.g. alpha) will plot all values of alpha.
quantiles	A vector with two values specifying the quantile values (lower and upper).
zeroline	If TRUE, a horizontal line at zero is drawn on the plot.
...	Additional arguments passed to plot.default

Author(s)

Ken Kellner <contact@kenkellner.com>.

Examples

```
#Analyze Longley economic data in JAGS
#Number employed as a function of GNP
#See ?jags for a more detailed example

#Get data
data(longley)
gnp <- longley$GNP
employed <- longley$Employed
n <- length(employed)
data <- list(gnp=gnp,employed=employed,n=n)

#Identify filepath of model file
modfile <- tempfile()

writeLines("
model{

  #Likelihood
  for (i in 1:n){

    employed[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha + beta*gnp[i]

  }

  #Priors
  alpha ~ dnorm(0, 0.00001)
  beta ~ dnorm(0, 0.00001)
  sigma ~ dunif(0,1000)
  tau <- pow(sigma,-2)

}
", con=modfile)

#Set parameters to monitor
params <- c('alpha','beta','sigma','mu')

#Run analysis

out <- jags(data = data,
            inits = NULL,
            parameters.to.save = params,
            model.file = modfile,
            n.chains = 3,
            n.adapt = 100,
            n.iter = 1000,
            n.burnin = 500,
```

```
            n.thin = 2)

#Examine output summary

out

#Generate whisker plots

#Plot alpha

whiskerplot(out,parameters=c('alpha'))

#Plot all values of mu

whiskerplot(out,parameters='mu')

#Plot a subset of mu

whiskerplot(out,parameters=c('mu[1]','mu[7]'))

#Plot mu and alpha together

whiskerplot(out,parameters=c('mu','alpha'))
```

Index

autojags, 2

densityplot, 4

jags, 5
jags.basic, 9
jags.View, 11
jagsUI *(jags)*, 5
jagsUI-class *(jags)*, 5
jagsUIbasic-class *(jags.basic)*, 9

plot.default, *12*, *15*
pp.check, 11

traceplot, 13

update, 14

View, *11*

whiskerplot, 15