JAGS Tutorial

1. What is JAGS?

JAGS stands for "**J**ust **A**nother **G**ibbs **S**ampler" and is a tool for analysis of Bayesian hierarchical models using Markov Chain Monte Carlo (MCMC) simulation. JAGS is an engine for running BUGS in Unix-based environments and allows users to write their own functions, distributions and samplers.
[Source: http://www-ice.iarc.fr/~martyn/software/jags/]

2. What is R2jags?

R2jags is an R package used to call JAGS from R.

3. Function Usage for R2jags Package

[Original full documentation at http://cran.r-project.org/web/packages/R2jags/R2jags.pdf]

---

autojags: Function for auto-updating jags until the model converges

---

Usage

```
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`	convergence criterion (default=1.1)
`n.update`	the max number of updates (default=2)
`refresh`	refresh frequency for progress bar (default=n.iter/50)
`progress.bar`	type of progress bar ("text" -- displayed on the R console; "gui" -- graphical progress bar in a new window; "none" -- progress bar is suppressed)

---

jags: 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) / n.sims)),
       n.sims = 1000, DIC=TRUE, working.directory=NULL, refresh = n.iter/50,
       progress.bar = "text")

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)
       / n.sims)), n.sims = 1000, DIC=TRUE, jags.path="", working.directory=NULL,
       clearWD=TRUE, refresh = n.iter/50)
```

Arguments

`data`	a vector or list of the names of the data objects used by the model
`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
`model.file`	file containing the model written in BUGS code
`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 beginning (default=n.iter/2; if n.burnin=0, jags() will run 100 iterations for adaption)
`n.thin`	thinning rate, must be a positive integer (default is max(1, floor(n.chains * (n.iter-n.burnin) / 1000)) which will only thin if there are at least 2000 simulations)
`n.sims`	approximate number of simulations to keep after thinning
`DIC`	logical; if TRUE (default), compute deviance, pD, and DIC; pD=var(deviance) / 2
`working.directory`	sets working directory (should be where model file is)
`jags.path`	directory that contains the jags executable (default="")
`clearWD`	indicates whether 'data.txt', 'inits[1:n.chains].txt', 'codaIndex.txt', 'jagsscript.txt', and 'CODAchain[1:nchains].txt' should be removed after jags finishes (default=TRUE)
`refresh`	refresh frequency for progress bar, default is n.iter/50
`progress.bar`	type of progress bar ("text" -- displayed on the R console; "gui" -- graphical progress bar in a new window; "none" -- progress bar is suppressed)

---

jags2bugs: Reads jags output files in CODA format 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 (should be where CODA files are)
`parameters.to.save`	character vector of the names of the parameters to save
`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 beginning (default=n.iter/2)
`n.thin`	thinning rate (default=2)
`DIC`	logical; if TRUE (default), compute deviance, pD, and DIC; pD=var(deviance) / 2

---

recompile: Function for recompiling previously saved rjags object

---

Usage

```
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=100)
`refresh`	refresh frequency for progress bar (default= n.iter/50)
`progress.bar`	type of progress bar ("text" -- displayed on the R console; "gui" -- graphical progress bar in a new window; "none" -- progress bar is suppressed)

traceplot: Displays a plot of iterations vs. sampled values for each variable in the chain.

Usage

```
traceplot(x, ...)
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 asked before each plot, see par(ask=.).
`col`	graphical parameter (see par)
`lty`	graphical parameter (see par)
`lwd`	graphical parameter (see par)
`...`	further graphical parameters

4. Example: week2.R

[Example source: http://cran.r-project.org/web/packages/R2jags/R2jags.pdf]

Programming steps:

1. Write a BUGS model in an ASCII file.
2. Open R.
3. Prepare the inputs for the jags function and run it.
4. The model will now run in JAGS. You will see progress in the R console.

Example code:

```
# R code:  rjags test
# Bayesian Data Analysis

library(rjags)
library(R2jags)

# An example model file is given in: /usr/local/lib/R/library/R2jags/model
# the plotted graphic is shown in your current directory by default, named Rplots.pdf
# load the file (Copy it to your R working directory if necessary)
      model.file <- system.file(package="R2jags", "model", "schools.txt")
# Let's take a look:
file.show(model.file)

# 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))
}
```

```
#=============#
# using jags  #
#=============#
jagsfit <- jags(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)

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

# if the model does not converge, update it!
jagsfit.upd <- update(jagsfit, n.iter=1000)
print(jagsfit.upd)

plot(jagsfit.upd)

# or auto update it until it converges! see ?autojags for details
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)
```