Report

Martina Le-Bert Heyl

2022-05-09

**My Github** <https://github.com/mlebertheyl>

**Github repository for this package** <https://github.com/mlebertheyl/myJpackage>

myJpackage website <https://mlebertheyl.github.io/myJpackage/>

# WHERE, WHAT, HOW …

## A brief (personal) motivational background.

Have you ever you ever been kin on learning something new, but don’t know where to start? Learning a new programming language or skill often feels like that (at least for me). There’s usually tons of information about it, but if you are new to it, this only makes the question of where to start even harder.

If you found where to start, you are on the right track, now the question gets more practical. I have found that first I need to understand what is the general structure when coding in a new environment, and how to do it myself.

Going through some basic or general examples does a lot in my learning process. However, the more specific a programming tool is, the more specific the examples you find. My motivation for the package was exactly giving focus on generating some examples that could help when learning a new coding skill.

## A more specific motivation for the problem.

Just Another Gibbs Sampler or better known as JAGS is a program for Bayesian graphical model analyses using Markov Chain Monte Carlo (MCMC) simulation. JAGS aims for compatibility with classic BUGS, Bayesian inference Using Gibbs Sampling, and other software such as R.

There are several packages to run JAGS using R, however we still need to code within BUGS programming language. This package aims to help understand the basic structure of JAGS models and data specification, so you can use the output code from the functions in this package or create your own using the output as a helper in your coding.

# WHAT IS THIS PACKAGE …

myJpackage is a collection of functions that will produce a code as output. It is the opposite of what we usually expect from an R function, where the output is a manipulation of the data in some way. Here, there are no data results, but actually a longer code. However, this is the code you can pass to your JAGS model. Also, the arguments for the functions have to do with the type of model you want to specify in JAGS.

# WHAT YOU WILL GET…

* General code for model specification.
* An idea on how to manipulate data within JAGS.
* JAGS parametrization for main distributions. JAGS parametrization is not the same as R!
* Built examples.
* Hopefully, a better understanding of the structure of JAGS code (or a useful reminder if you forget later on).

# WORKFLOW…

You can use the package in any way you want, but if you are new to JAGS I would recommend the following trying these functions first.

* jags\_init()
* jags\_examples()
* jags\_model()
* jags\_runjags()

# REMINDER

You need to install JAGS on your computer and then install the runjags library. The JAGS version will depend on your computer operating system, jags\_init() will be of help.

If you want to know more about JAGS, and not about the american football team Jacksonville Jaguars, use “jags bayes” on your Google search.

# EXAMPLE

This is an example using the output code from jags\_example function, and the following implementation using the runjags library.

library(myJpackage)  
library(runjags)

## Warning: package 'runjags' was built under R version 4.1.2

jags\_examples("model1", runme = TRUE)

## model1 <- "model {  
## for (i in 1:N) {  
## eta[i] <- a + b \* x[i]  
## logit(prob[i]) <- eta[i]  
## y[i] ~ dbern(prob[i])  
## }  
## a ~ dnorm(0, 0.01)  
## b ~ dnorm(0, 0.01)  
## }"  
## n <- 100  
## x <- rnorm(n)  
## eta <- 1 + x  
## prob <- 1/(1+exp(-eta))  
## y <- rbinom(n, size = 1, prob = prob)  
## library(runjags)  
## r <- combine.mcmc(run.jags(model = model1,  
## data = list(y = y, x = x, N = n),  
## monitor = c("prob[1]", "a", "b"),  
## n.chains = 4,  
## sample = 10^5,  
## method = "parallel"))  
## summary(r)

model1 <- "model {  
 for (i in 1:N) {  
 eta[i] <- a + b \* x[i]  
 logit(prob[i]) <- eta[i]  
 y[i] ~ dbern(prob[i])  
 }  
 a ~ dnorm(0, 0.01)  
 b ~ dnorm(0, 0.01)  
}"  
  
n <- 100  
x <- rnorm(n)  
eta <- 1 + x  
prob <- 1/(1+exp(-eta))  
y <- rbinom(n, size = 1, prob = prob)  
  
r <- combine.mcmc(run.jags(model = model1,  
 data = list(y = y, x = x, N = n),  
 monitor = c("prob[1]", "a", "b"),  
 n.chains = 4,  
 sample = 10^5,  
 method = "parallel"))

## Warning: No initial values were provided - JAGS will use the same initial values  
## for all chains

## Warning: You attempted to start parallel chains without setting different PRNG  
## for each chain, which is not recommended. Different .RNG.name values have been  
## added to each set of initial values.

## Calling 4 simulations using the parallel method...  
## Following the progress of chain 1 (the program will wait for all chains  
## to finish before continuing):  
## Welcome to JAGS 4.3.1 (official binary) on Mon May 9 22:58:53 2022  
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY  
## Loading module: basemod: ok  
## Loading module: bugs: ok  
## . . Reading data file data.txt  
## . Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 100  
## Unobserved stochastic nodes: 2  
## Total graph size: 505  
## . Reading parameter file inits1.txt  
## . Initializing model  
## . Adapting 1000  
## -------------------------------------------------| 1000  
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%  
## Adaptation successful  
## . Updating 4000  
## -------------------------------------------------| 4000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 100000  
## -------------------------------------------------| 100000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 0  
## . Deleting model  
## .   
## All chains have finished  
## Simulation complete. Reading coda files...  
## Coda files loaded successfully  
## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 3 variables....  
## Finished running the simulation

summary(r)

##   
## Iterations = 5001:405000  
## Thinning interval = 1   
## Number of chains = 1   
## Sample size per chain = 4e+05   
##   
## 1. Empirical mean and standard deviation for each variable,  
## plus standard error of the mean:  
##   
## Mean SD Naive SE Time-series SE  
## prob[1] 0.8146 0.04833 7.641e-05 0.0001074  
## a 0.8055 0.25510 4.033e-04 0.0005354  
## b 1.4934 0.34284 5.421e-04 0.0007285  
##   
## 2. Quantiles for each variable:  
##   
## 2.5% 25% 50% 75% 97.5%  
## prob[1] 0.7117 0.7835 0.8176 0.8491 0.900  
## a 0.3181 0.6312 0.8010 0.9738 1.320  
## b 0.8647 1.2553 1.4788 1.7152 2.205

# FINAL COMMENTS

The puropose was to make a fun package, specially by making it friendly and also useful. The main idea was to gain a better understanding of the structure of JAGS code, and to use it as a reminder later on, since JAGS and R structure and code differ a little.

Details of the package’s functions are available on myJpackage website <https://mlebertheyl.github.io/myJpackage/>, and details on the package’s construction can be found on <https://github.com/mlebertheyl/myJpackage>.