Bayes Workshop Lab Day 1

Dr. Sarah Hunter

5/18/2020

Overview

Today's Lab will be devoted to installing the software necessary for Bayesian Analysis. I will be teaching in JAGS, which stands for Just Another Gibbs Sampler. JAGS is a software that does Baeysian analysis through R. Therefore, some knowledge of R is preferred. I will also stick around during this Lab for anyone who needs a refresher course on R. My website contains some help files that cover the basics of R: https://www.sarahghunter.com/teaching.

Installing R

For those of you who do not have R installed on your device, or have not updated in a while, you can find the latest version of R here: https://cran.r-project.org. Having the most up to date version of R is important for many of the packages needed in later stages of Bayesian Analysis. The current version of R is 4.0.0. To either update an existing version or install R for the first time, go to the previous link and click on the appropriate version (Mac v. PC) and install the latest version.

Installing JAGS

JAGS is a free program built by Martyn Plummer. You can find the download site here: https://sourceforge.net/projects/mcmc-jags/files/. Click on the green button that says "Download Latest Version". Please remember to also download and read the User Manual, under "Manuals".

Running JAGS through R

JAGS can run seamlessly through R with the help a a couple R packages. Part of today's Lab is also installing those packages.

The JAGS programming language is very similar to R, however there some small differences, especially with the names of distributions. However, it is similar enough to R that anyone with some R training can understand JAGS easily. In order to run JAGS through the R interface, you should download the packages `rjags` and `R2jags` using the usual `install.pacakges()` command.

```
#install.packages("rjags")
#install.packages("R2jags")
```

To make sure everything is functioning correctly, please run the code in the following section. Do not worry about understanding it currently. Tomorrow, we will go over this code in more detail.

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
## Loaded modules: basemod,bugs
```

```
library(R2jags)
```

```
##
## Attaching package: 'R2jags'
## The following object is masked from 'package:coda':
##
##     traceplot
```

```
#load data

dat<-list( vote = c( 57.1, 16.8, 86.1, 40.8, 90.3, 35.5, 96.7, 44.6, 9.1,
                     97.6, 79.2, 31.2, 70.45, 49.53, 53.16),
           gnp = c( 2.42, 0.07, 0.26, 1.42, 3.11, 2.88, 4.18, 2.33, -1.38,
                    3.95, 1.91, 1.46, 1.85, 2.52, 1.47),
           approval = c( 39, 32, 69, 49, 74, 40, 56, 45, 21, 52, 51, 32, 57,
                         59, 47), N = 15 )

data<-as.data.frame(dat)

#Model

linear.mod.jags<-function(){
    for(i in 1:N){
        vote[i]~dnorm(mu[i], tau)
        mu[i]<-alpha+beta1*gnp[i] + beta2*approval[i]
    }

    alpha~dnorm(0, .1)
    beta1~dunif(-100,100)
    beta2~dunif(-100,100)
    tau~dgamma(.5,.5)

}

vote<-data$vote
gnp<-data$gnp
approval<-data$approval

my.data<-list("vote", "gnp", "approval", "N")
my.data<-list(vote=data$vote, gnp=data$gnp, approval=data$approval,
              N=length(data$vote))

my.params<-c("alpha", "beta1", "beta2")


#Fit the model
my.model<-jags(data=my.data, inits=NULL, my.params, n.chains=2, n.iter=9000,
               n.burnin=1000, model.file=linear.mod.jags)
```

```
## module glm loaded
## Compiling model graph
```

```
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 4
##    Total graph size: 99
##
## Initializing model
```

```
print(my.model)
```

```
## Inference for Bugs model at "/var/folders/dx/pkx9cyj1089843ljm_jzj3pm0000gn/T//RtmpN3RWTM/modelce816
##  2 chains, each with 9000 iterations (first 1000 discarded), n.thin = 8
##  n.sims = 2000 iterations saved
##          mu.vect sd.vect    2.5%     25%     50%     75%   97.5%  Rhat n.eff
## alpha     -0.876   3.071  -6.780  -2.942  -0.863   1.188   5.180 1.003  2000
## beta1      7.164   3.336   0.602   4.981   7.240   9.291  13.598 1.002  1400
## beta2      0.948   0.170   0.618   0.836   0.951   1.061   1.287 1.002  1000
## deviance 125.724   2.572 122.351 123.924 125.103 126.872 131.925 1.001  2000
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 3.3 and DIC = 129.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

If the above code runs without errors and you manage to get results, everything is functioning correctly. Do not worry if your results are slightly different than mine.

Next time: Bayesian Linear Models