Introduction to Bayesian Statistics

Installing JAGS

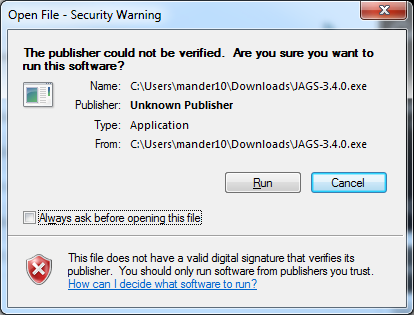
**Downloading JAGS**

JAGS is free to download at the following website

<https://sourceforge.net/projects/mcmc-jags/files/>

Select the JAGS file folder, the release of JAGS you wish to install (4.x, 3.x, etc), the platform you’ll be running JAGS on (Windows, Mac, Linux), and finally the version of the release you want (JAGS 3.4.0.exe is the current DOD approved version).

After the download, select ‘Run’ and the follow the on screen directions for the installation process.



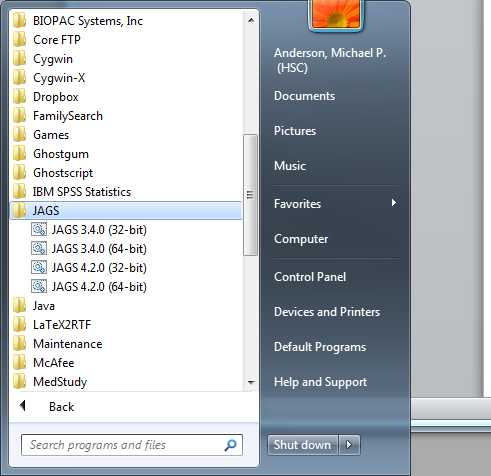
This will launch the JAGS Set-up Wizard that requires you to consent to the license agreement, choose which computer users may access JAGS along with an installation location and finally choose 32 and/or 64 bit versions for installation.



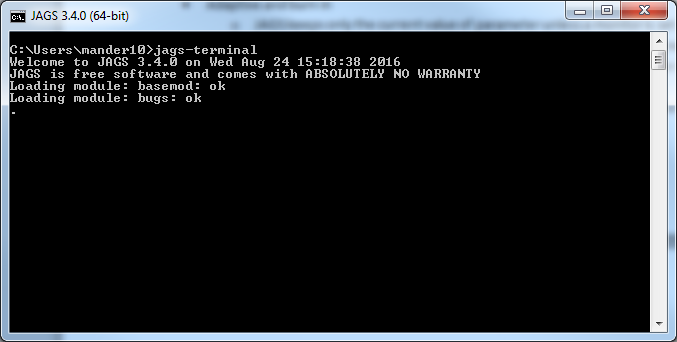
Choosing the defaults will create a JAGS folder in your ‘All Programs’ menu with both 32 and 64 bit versions.

**Open JAGS**

If you have selected the defaults while installing JAGS, both the 32-bit and 64-bit versions will be available in your ‘All Programs’ folder.



Launching the 64 bit version of JAGS will open an command console that looks like the following.



Notice the ‘.’ on the far left portion of the screen. This means JAGS is waiting for a command to execute. Commands may be issued here identifying files for models, data, initial values, and parameters to monitor. You may also issue the coda command to output the MCMC samples.

There are 5 steps JAGS uses to obtain MCMC samples.

1. Define model and data
2. Compile model and data
3. Initialize the MCMC chains
4. Adaptive and burn in phase of MCMC
5. Monitor the parameters

NOTE: All else (convergence diagnostics, model criticism, summary samples, plotting, etc) must be done using other software. We will plan to use R to do all of this other stuff.

**Define the Model and Data**

JAGS uses BUGS like notation to define the model to be used in the MCMC. For the refueling example our model statement would look something like

model

{

#likelihood

for(i in 1:24)

{

refuel[i]~dbern(disconnect.prob)

}

#prior

disconnect.prob~dbeta(alpha,beta)

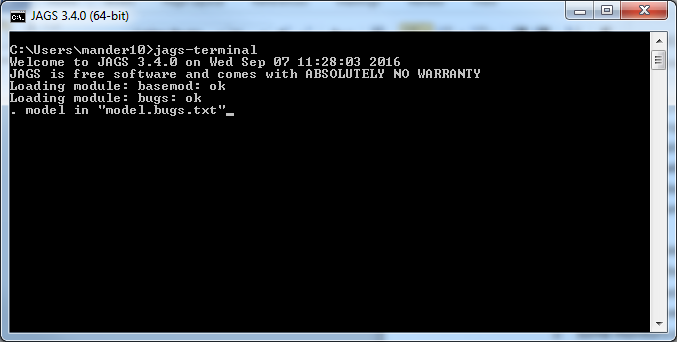
alpha<-1

beta<-1

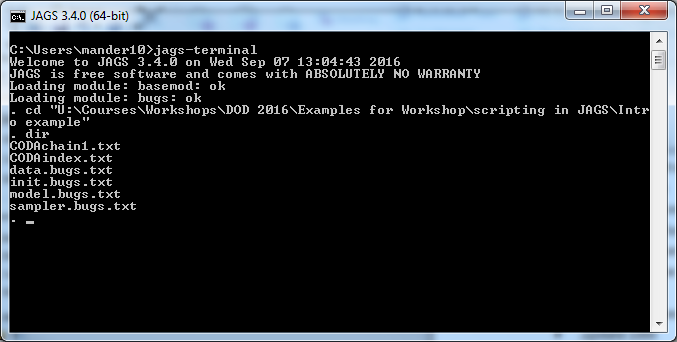
}

A few points of interest in the above model statement are: 1. the for loop which assigns a Bernoulli distribution to each of the 24 observations in the data vector named ‘refuel’, 2. the # which works as a comment much as it would in R, and 3 the use of the ~ vs the <-. The ~ indicates a stochastic relationship whereas the <- indicates a deterministic relationship. As we saw in the examples earlier, all we need are the prior and the likelihood and MCMC takes care of the rest.

We will save this model statement to a file (naming it something like ‘model.bugs.txt’ using notepad will work). This model can be uploaded to JAGS using ‘model in “file.name.txt”’ command.



You may need to change the directory JAGS using ‘.cd “path\to\file”’ in order to access the files you’ve saved. Also .pwd will print the current working directory and .dir will list the files in the current directory.

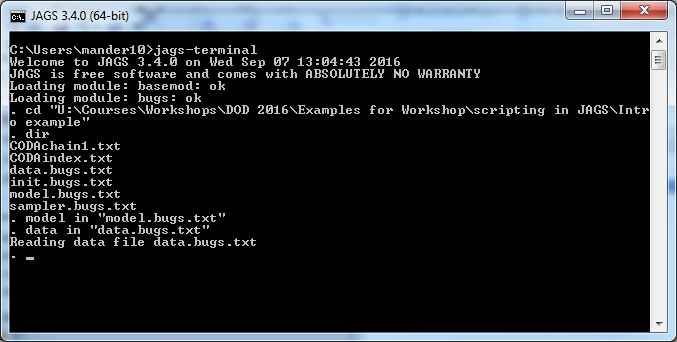


Next you will need to specify the data

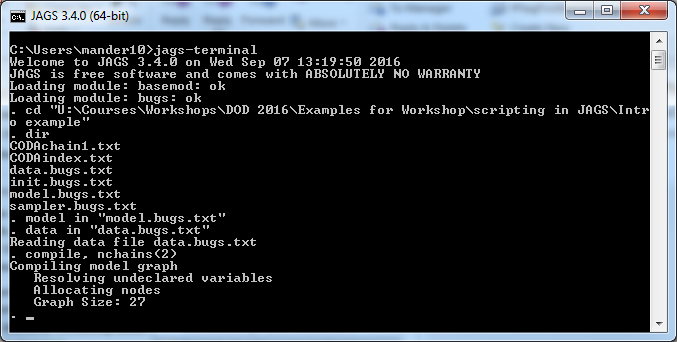
#Data

sex<-c(0,0,1,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,1,0,1,0,0,1)

Notice that the data is specified as a vector here. Data. Frames will work as well but lists will not. We will save these data to a file (something like “data.bugs.txt”) to be read into JAGS.



Next you will compile the data using the .compile statement. You can also specify the number of chains to be used in the MCMC at this point.



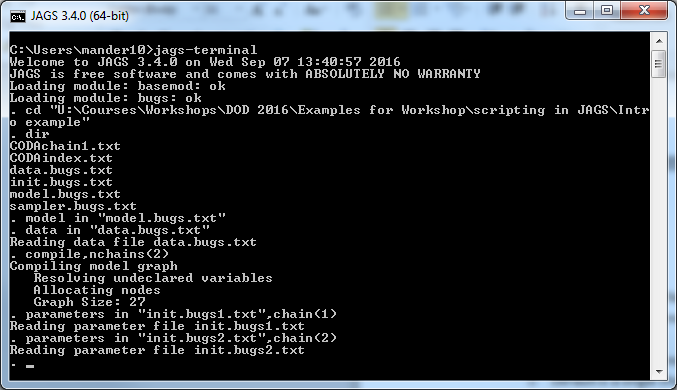
Now the initial values for the parameters need to be specified. We only have one parameter for this example so we can do this with the syntax

#Initial Values

disconnect.prob<-0.65

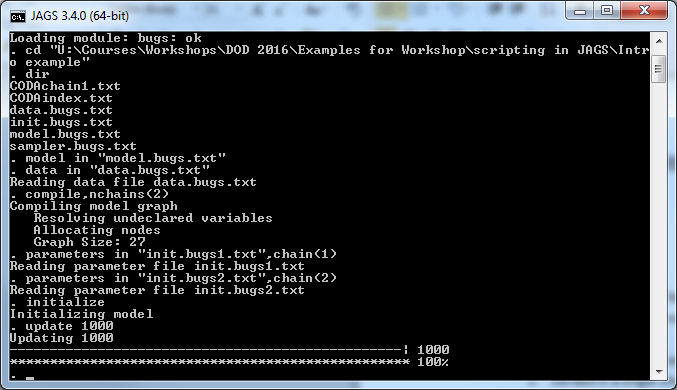
#Initial Values

disconnect.prob<-0.10

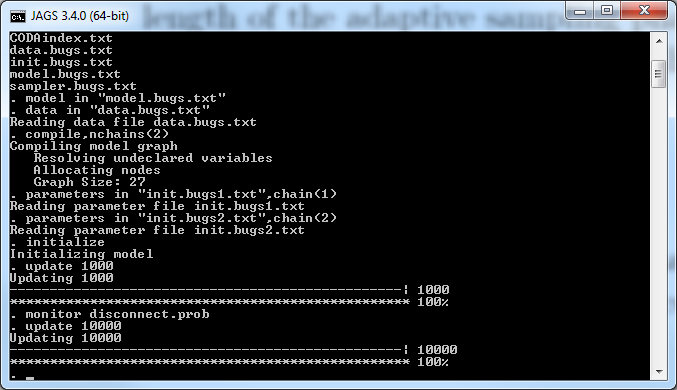
The choice of initial values is totally up to you. In this example they just need to be something between 0 and 1. If we want to check to see if the model converged well, we might start the two chains at the extremes and see if they get to the same place. As with the model and data, we will save this to a file (suppose we call this “init.bugs1.txt” and “init.bugs2.txt” for the initial values of chain 1 and 2 respectively) to be called in the current JAGS session. 

The ‘parameters in “init.bugs1.txts”,chain(1)’ command specifies the initial value for chain 1 while ‘parameters in “init.bugs2.txt”,chain(2)’ specifies the initial value for chain 2.

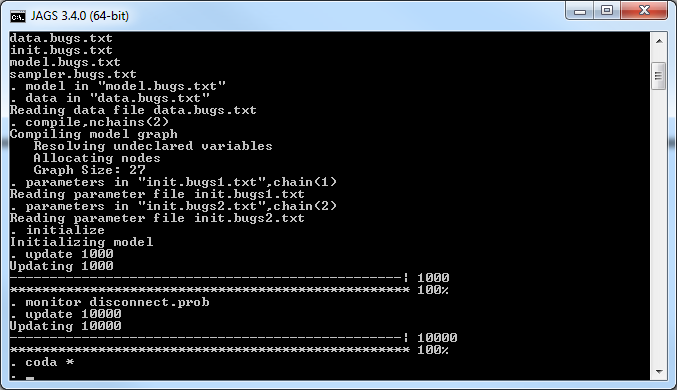
Now we can initialize the model using .initialize and then obtain burn-in updates using .update 1000 (doesn’t need to be 1000, just needs to be long enough that the MCMC has reached a stationary distribution.



Now that the burn-in period is out of the way, we can start monitoring the parameters of interest with the ‘.monitor’ statement. In this example we have only one parameter so we tell JAGS to monitor that parameter while giving us 10000 more updates for the 2 chains.



Now the MCMC process is complete and we need to output the samples from the posterior distribution. This can be done using the ‘.coda \*’ statement. This will dump all the posterior samples for all monitored variables to the current directory as a CODAindex.txt file. These forms can be read by the coda package of R.



Now looking in the directory will show additional files with the prefix name of “CODA…” I have combined information from the three files below including the first 20 posterior samples from both chains.

|  |  |
| --- | --- |
| **CODAindex** | disconnect.prob 1 10000 |
| **CODAchain1** | **CODAchain2** |
| 1001 0.259635  1002 0.354686  1003 0.292825  1004 0.285003  1005 0.238503  1006 0.283204  1007 0.267325  1008 0.30765  1009 0.330016  1010 0.418721  1011 0.233574  1012 0.292361  1013 0.24749  1014 0.190346  1015 0.30429  1016 0.270001  1017 0.281775  1018 0.171369  1019 0.438506  1020 0.123394 | 1001 0.313101  1002 0.138449  1003 0.373837  1004 0.221288  1005 0.431186  1006 0.260662  1007 0.257465  1008 0.384764  1009 0.371788  1010 0.448239  1011 0.409164  1012 0.438611  1013 0.281664  1014 0.295716  1015 0.182753  1016 0.399483  1017 0.29923  1018 0.328038  1019 0.112449  1020 0.459609 |