Single Season Example of MSOM

Sterling Cole

March 21, 2018

Summary of Objectives

This document is meant to be an introductory guide to multi-species occupancy modelling (MSOM). I included a mix of code from the Kery & Royle's Advanced Hierarchical Modelling book https://www.mbr-pwrc.usgs.gov/pubanalysis/keryroylebook/ and modified a JAGS model developed by Tingley et al. 2016. Data and JAGS code for that model are stored here: https://datadryad.org/resource/doi:10.5061/dryad.871pc. There's plenty more explanation of these types of model in Chap. 11 of the AHM book.

Housekeeping

Now that we're working on Bayesian modelling in BUGS you should download and install the latest version of JAGS. It's available here: https://sourceforge.net/projects/mcmc-jags/files/JAGS/

Now for some code

The first step is loading in the jagsUI package which I prefer for running Bayesian models. The package allows for relatively easy manipulation of output and allows you to easily run a model in parallel on multiple cores (if you're so lucky to have computer with that capability). This can cut run-time by a factor of 3. If you don't have it installed, just run the commented out code too. Also load up the AHMbook package to load up some dependencies.

```
# install.packages("jagsUI")
# install.packages("AHMBook")
library(jagsUI)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'jagsUI'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

```
library(AHMbook)
```

```
## Loading required package: unmarked
```

```
## Loading required package: reshape
```

```
## Loading required package: parallel
```

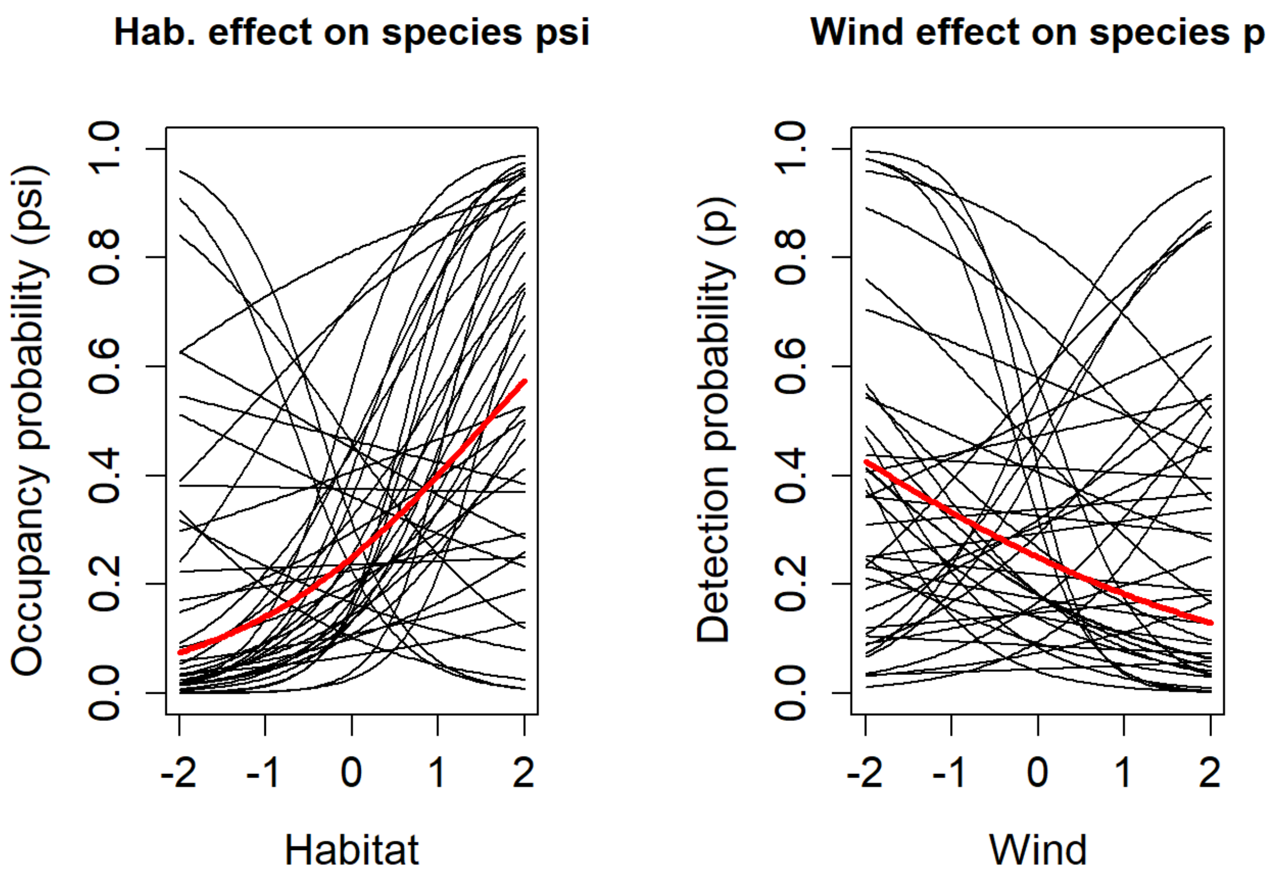
```
## Loading required package: Rcpp
```

```
# Download the custom simulation function
download.file(url="https://raw.githubusercontent.com/jsc329/MSOM-Example/master/Rev
ised%20community%20function.R", destfile="simfunc.R")
# This is saved in your working directory, use getwd() to figure out yours
# Load the function into the workspace
source("simfunc.R")
```

We have a function - simCommEdit - for generating state covariates that influence the occupancy of sites, and detection covariates that influence the detection probability of individual species. It also makes a nice tidy dataset with simulated detections of species given user specified relationships to state and detection covariates. So let's simulate our first dataset!

First simulation

```
simdata <- simCommEdit(type="det/nondet", nsite=30, nrep=3, nspec=40,
           mean.psi=0.25, sig.lpsi=1, mu.beta.lpsi=0.7, sig.beta.lpsi=1,
           mean.p=0.25, sig.lp=1, mu.beta.lp=-0.4, sig.beta.lp=1, show.plot = T)
```



So here's a quick rundown of all the arguments in the function:

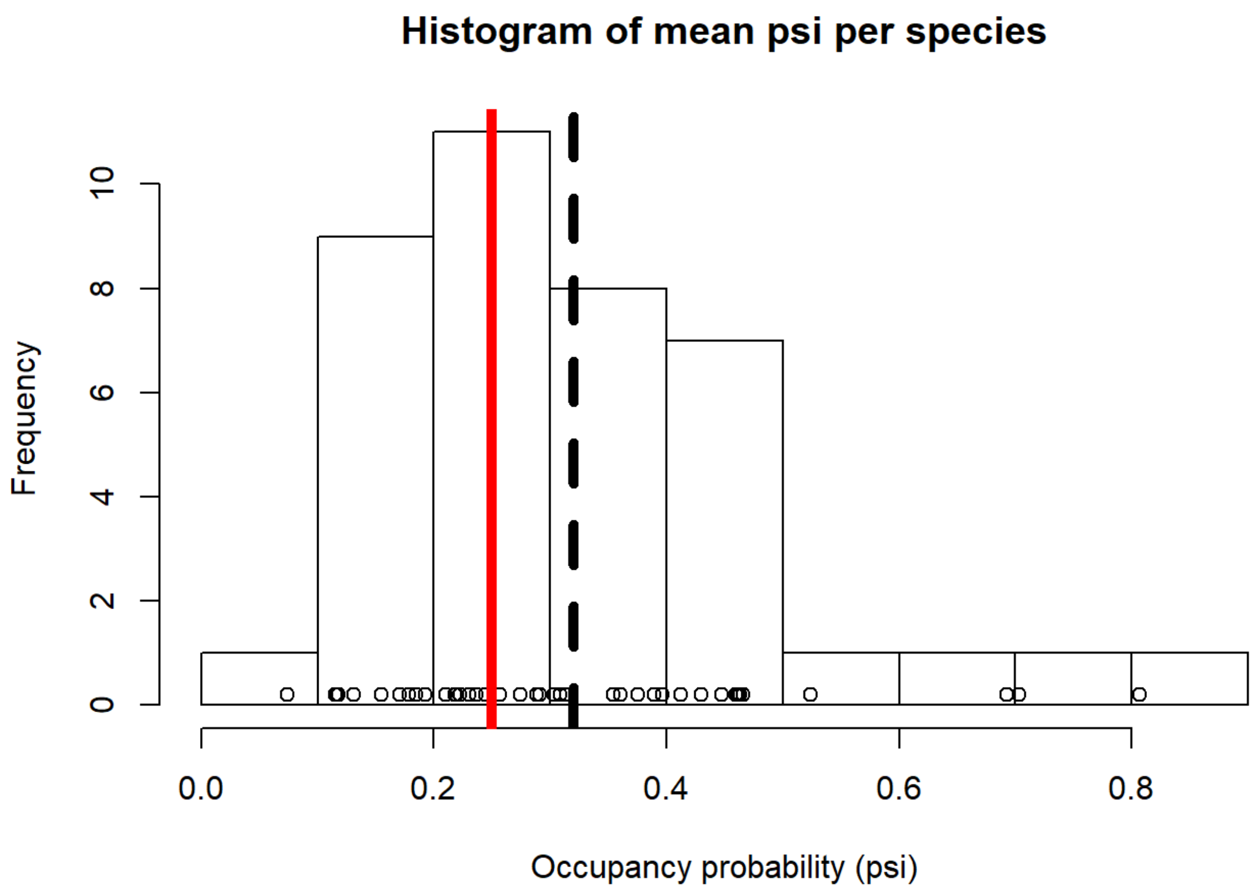
- type = let's the function know we want an occurrence output
- nsite = number of sampled sites
- nrep = number of visits to each site
- nspec = the size of the species community (this will likely be smaller in the output)

- mean.psi = the mean occupancy probabilty across all species
- sig.lpsi = the sd of occupancy probability for all species (how different psi is from species to species)
- mu.beta.lpsi = the mean effect of habitat on occupancy probability across all species
- sig.beta.lpsi = the sd of effect of habitat on occupancy probability (again how effect varies species to species)
- mean.p = the mean detection probability across all species
- sig.lp = the sd of detection probability (how p varies species to species)
- mu.beta.lp = the mean effect of wind on detection probability across all species
- sig.beta.lp = the sd of detection probability (how p varies species to species)
- show.plot = gives summary of output in graphical form. Plots of psi versus habitat, and p versus wind

First lesson

The MSOM hinges on the idea that the occupancy probability and detection probability come from some underlying distribution. For lack of a better idea, we say that they come from a normal distribution. If you run the simulation function above you'll see that occupancy probability increases and decreases with amount of habitat depending on the species, but in general across all species occupancy probability increases. We call this the community effect and it is estimated by a hyperparameter (a parameter that governs other parameters). You can think of this as the average effect across all species. The occupancy probability for each species also comes from a overall distribution governed by a hyperparameter. A nice illustration of this is below:

```
hist(apply(simdata$psi, 2, mean), ylab="Frequency", xlab="Occupancy probability (psi)", main="Histogram of mean psi per species")
points(x=apply(simdata$psi, 2, mean), y=rep(0.2, simdata$nspec))
abline(v=mean(apply(simdata$psi, 2, mean)), lwd=5, lty=2)
abline(v=simdata$mean.psi, lwd=5, lty=1, col="red")
```



We are taking the average occupancy probability across all sites using apply(), then plotting a histogram of that data. Depending on how your simulation ran you'll get a histogram that approximates a normal distribution. Those dots at the bottom of the plot are the known occupancy probabilities for each species in the model. The dotted black line is the mean of that data, and the solid red line the mean value of psi we gave to the simulation function. You can think of the dotted black line as the average occupancy probability for the community of birds, with indvidual species closer or further from that mean. The difference you see between the two bars is due to the different responses of birds species to habitat. Going to jump ahead a little and say, we know that psi[i,k] = plogis(beta0[k] + beta1[k] X habitat[i]), so psi is the sum of the average occupancy probability for each species *k* (beta0[k]) and occupancy probability for species *k* across all habitat values at each site *i*. We can use this information to sort-of guess at what the occupancy probability is for species with much fewer observations, because we known the average effect (or hyperparameter).

Organizing the simulation output

Unfortunately for you (and me) the output from the simulation function isn't exactly conducive to feeding into JAGS. So we're going to do a little data organization. We're going to store our occurence data in an array. Since you're probably a pro user of R you already know what a matrix is, it has x rows, and y columns. An array is kind-of like that except on steroids. You can have one two, three, four, five, however many dimensions you'd like. We're going to stick with just 3d here. If a matrix is like a sheet of paper with a spreadsheet printed on it, a 3d array is like a book with all of those sheets of paper. In this case the rows are sites, the columns are visits to sites, and the pages are the species. So for instance if we called our array ydata, then ydata[,,1] would essentially output a matrix with occurence data for species 1. Now lets do it:

```
ytemp <- simdata$y.obs # Store the output of the simulation here

# Make an empty array with dimensions [site, visit, species]
ydata <- array(data=NA, dim=c(simdata$nsite, simdata$nrep, dim(simdata$y.obs)[3]) )


  for (rep in 1:simdata$nrep){ # Step through all visits
    for (spec in 1:dim(simdata$y.obs)[3]){ # Step through all species

      ydata[ , rep, spec] <- unlist(ytemp[, rep, spec]) # All sites, visit number
"rep", species number "spec"

    }
  }

# Check the full array
# "all" checks to see if all values are TRUE
# Make sure that you stored everything correctly in the array
all(ytemp[,,]==ydata[,,])
```

```
## [1] TRUE
```

```
# Do a similar thing with the habitat and wind data
# We don't need an array here because we only have one set of covariates
# that are shared across all species
windcov <- as.matrix(simdata$wind, byrow=T, row=simdata$nsite, col=simdata$nrep)
habcov <- as.vector(simdata$habitat)

# Check to make sure things match again
all(windcov==simdata$wind)
```

```
## [1] TRUE
```

```
all(habcov==simdata$habitat)
```

```
## [1] TRUE
```

Finally running some JAGS code

Now that we have all our data ducks in a row we can do some fun stuff, like running the model. We typically would use scale() to standardize covariates to facilitate faster convergence, but lucky for use the simulation function spits our covariates pre-standardized. Now we are going to specify starting value for true occupancy state *Z*. We can't know for sure whether an individual is at a particular site so we call *Z* a latent variable, something we aren't capable of observing. To help the model pick start at a reasonable point we'll give it some guess as to whether a site was really occupied by saying that any site with at least 1 observation was probably occupied. We use apply for this. We'll also download the jags model from github. You can take a peak at it by finding the PyroModel.R file saved in your working directory after you run the following code.

```
Z <- apply(ydata, c(1,3), max)
effort <- c(1, 0, 0) # This shouldn't make a difference, but I left it in because
```

```
# if you want to model your data you might consider a dummy variable indicating
# the length of the survey period, First was 3 mins, and the last two were only 2 mins
# The simulation function currently does not simulate an effect of this

# Download the custom simulation function
download.file(url="https://raw.githubusercontent.com/jsc329/MSOM-Example/master/PyroModel.R", destfile="PyroModel.R")
# This is saved in your working directory, use getwd() to figure out yours

# This is where we store our initial values
# We keep them in a function because that's how jags like them
# You can actually specify starting values for any unobserved
# values in jags, by adding to this list
inits <- function() { list(Z=Z)}

# Here's where we add in everything we have observed for use in the model
# including the number of species, sites, and visits
# This are used for loops inside of the model
data <- list(hab=habcov, wind=windcov, y=ydata, effort=effort,
             n.species=dim(ydata)[3], n.sites=dim(ydata)[1],
             n.visits=dim(ydata)[2])

nc <- 3 # number of chains to run, typically 3
ni <- 5000 # number of iterations to run, can vary alot depending on how well your model is converging
nb <- 1000 # number of burn in iterations
# Since jags typically starts with a guess of what a parameter estimate should be
# we typically discard a certain number of iterations at the beginning
# because jags is searching and hasn't yet found what it thinks is the best estimate
# we'll see more of and example of this in the output
nt <- 4 # How many of the iterations to save
# With 1 we save all iterations, but with 3 we'd save every 3rd iteration
# This can help we condensing down a large output
na <- 500 # The magic number of adaptive iterations
# during these runs the model tunes itself to optimize finding the parameter estimates
# in the quickest time

# speaking of parameter estimates, we specify what we want jags to save explicitly
# MT seems to keep the names of his parameters short
# What we're saving are the estimates of parameters that affect detection probability *a*
# and those that affect occupancy probability *b*
# a[i, 1] the intercept for detection probability for species *i* for instance
params <- c("a", "b")

# Here's where the rubber meets the road
# can kinda think of this as the lm() function for Bayesian modelling
# I turned parallel to TRUE to speed up the model fitting process
# But I'd recommend turning it to F so you can see that pretty progress bar
# With parallel=T should take 3-5 minutes
mod <- jags(data=data, inits=inits, model.file="PyroModel.R", n.iter=ni,
            n.chains=nc, n.adapt=na, parameters.to.save = params, parallel=T,
            n.burnin=nb, n.thin=nt)
```

```
##
## Processing function input.......
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
# Take a look at the brief summary output
mod
```

```
## JAGS output for model 'PyroModel.R', generated by jagsUI.
## Estimates based on 3 chains of 5000 iterations,
## adaptation = 500 iterations (sufficient),
## burn-in = 1000 iterations and thin rate = 4,
## yielding 3000 total samples from the joint posterior.
## MCMC ran in parallel for 3.943 minutes at time 2018-03-22 22:17:41.
##
##              mean    sd    2.5%     50%   97.5% overlap0     f  Rhat
## a[1,1]     -1.929 0.687  -3.277  -1.935  -0.567    FALSE 0.997 1.007
## a[2,1]      0.077 0.401  -0.717   0.076   0.859     TRUE 0.574 1.001
## a[3,1]     -1.348 0.629  -2.619  -1.341  -0.125    FALSE 0.985 1.000
## a[4,1]     -0.643 0.563  -1.727  -0.638   0.401     TRUE 0.871 1.000
## a[5,1]     -1.627 0.732  -3.069  -1.625  -0.179    FALSE 0.989 1.000
## a[6,1]      0.043 0.555  -1.076   0.047   1.120     TRUE 0.538 1.003
## a[7,1]     -1.575 0.687  -2.951  -1.554  -0.257    FALSE 0.988 1.010
## a[8,1]     -1.851 0.676  -3.170  -1.860  -0.487    FALSE 0.996 1.007
## a[9,1]     -0.090 0.359  -0.776  -0.087   0.617     TRUE 0.599 1.002
## a[10,1]    -1.451 0.794  -2.964  -1.460   0.126     TRUE 0.964 1.000
## a[11,1]    -1.950 0.641  -3.250  -1.933  -0.725    FALSE 0.999 1.010
## a[12,1]    -0.526 0.400  -1.318  -0.534   0.259     TRUE 0.904 1.001
## a[13,1]    -1.661 0.485  -2.616  -1.664  -0.699    FALSE 1.000 1.002
## a[14,1]    -1.300 0.569  -2.414  -1.313  -0.166    FALSE 0.988 1.000
## a[15,1]    -0.680 0.471  -1.592  -0.690   0.275     TRUE 0.930 1.000
## a[16,1]    -0.841 0.468  -1.747  -0.846   0.090     TRUE 0.961 1.001
## a[17,1]    -1.082 0.657  -2.358  -1.064   0.188     TRUE 0.955 1.001
## a[18,1]    -1.188 0.565  -2.311  -1.184  -0.111    FALSE 0.984 1.003
## a[19,1]    -2.088 0.862  -3.761  -2.105  -0.422    FALSE 0.992 1.015
## a[20,1]    -1.012 0.625  -2.181  -1.038   0.236     TRUE 0.943 1.002
## a[21,1]    -2.074 0.885  -3.804  -2.068  -0.258    FALSE 0.985 1.006
## a[22,1]    -0.977 0.554  -2.047  -0.967   0.115     TRUE 0.963 1.001
## a[23,1]    -1.338 0.487  -2.323  -1.338  -0.384    FALSE 0.997 1.001
## a[24,1]    -1.621 0.584  -2.769  -1.619  -0.473    FALSE 0.999 1.000
## a[25,1]    -0.440 0.405  -1.234  -0.442   0.346     TRUE 0.861 1.001
## a[26,1]    -1.022 0.579  -2.119  -1.028   0.144     TRUE 0.955 1.000
## a[27,1]    -1.341 0.491  -2.307  -1.346  -0.372    FALSE 0.996 1.001
## a[28,1]    -1.459 0.599  -2.641  -1.459  -0.306    FALSE 0.993 1.003
## a[29,1]    -0.099 0.345  -0.766  -0.096   0.598     TRUE 0.612 1.002
## a[30,1]    -1.609 0.946  -3.477  -1.609   0.229     TRUE 0.959 1.006
```

```
## a[31,1]     -1.466  0.717   -2.903   -1.458   -0.052    FALSE 0.980 1.004
## a[32,1]     -1.368  0.826   -2.974   -1.376    0.264     TRUE 0.947 1.006
## a[33,1]      0.459  0.559   -0.651    0.463    1.573     TRUE 0.795 1.002
## a[34,1]     -2.065  0.888   -3.797   -2.096   -0.300    FALSE 0.989 1.009
## a[35,1]      0.370  0.288   -0.172    0.368    0.968     TRUE 0.901 1.001
## a[36,1]     -1.334  0.788   -2.822   -1.337    0.234     TRUE 0.956 1.000
## a[37,1]     -1.246  0.471   -2.196   -1.240   -0.335    FALSE 0.996 1.001
## a[38,1]     -1.630  0.598   -2.812   -1.645   -0.451    FALSE 0.996 1.003
## a[39,1]     -1.627  0.549   -2.764   -1.613   -0.547    FALSE 0.998 1.001
## a[40,1]     -2.087  0.892   -3.760   -2.097   -0.320    FALSE 0.989 1.010
## a[1,2]      -0.669  0.649   -2.091   -0.615    0.522     TRUE 0.866 1.000
## a[2,2]      -1.523  0.504   -2.584   -1.493   -0.605    FALSE 1.000 1.001
## a[3,2]       0.546  0.540   -0.456    0.519    1.713     TRUE 0.851 1.001
## a[4,2]      -1.245  0.631   -2.614   -1.187   -0.134    FALSE 0.989 1.000
## a[5,2]      -0.614  0.517   -1.651   -0.604    0.353     TRUE 0.883 1.000
## a[6,2]       0.451  0.549   -0.568    0.432    1.586     TRUE 0.800 1.001
## a[7,2]       1.158  0.700   -0.079    1.101    2.670     TRUE 0.964 1.001
## a[8,2]      -0.070  0.559   -1.229   -0.067    0.985     TRUE 0.546 1.000
## a[9,2]       0.524  0.363   -0.172    0.520    1.266     TRUE 0.930 1.001
## a[10,2]     -0.091  0.579   -1.179   -0.108    1.087     TRUE 0.577 1.000
## a[11,2]     -0.738  0.637   -2.121   -0.697    0.419     TRUE 0.891 1.000
## a[12,2]     -0.186  0.357   -0.889   -0.184    0.498     TRUE 0.701 1.000
## a[13,2]      0.584  0.422   -0.210    0.574    1.427     TRUE 0.926 1.001
## a[14,2]      0.492  0.496   -0.394    0.461    1.562     TRUE 0.849 1.000
## a[15,2]      1.092  0.557    0.160    1.052    2.314    FALSE 0.992 1.001
## a[16,2]     -0.251  0.381   -0.995   -0.240    0.471     TRUE 0.744 1.001
## a[17,2]      0.646  0.579   -0.441    0.619    1.795     TRUE 0.876 1.001
## a[18,2]     -0.911  0.542   -2.064   -0.871    0.062     TRUE 0.966 1.000
## a[19,2]     -0.369  0.693   -1.825   -0.357    0.970     TRUE 0.711 1.004
## a[20,2]      0.976  0.595   -0.080    0.930    2.290     TRUE 0.966 1.001
## a[21,2]     -0.362  0.683   -1.692   -0.341    0.919     TRUE 0.697 1.000
## a[22,2]     -0.154  0.494   -1.206   -0.140    0.785     TRUE 0.620 1.001
## a[23,2]     -0.449  0.495   -1.503   -0.435    0.478     TRUE 0.813 1.000
## a[24,2]      0.650  0.488   -0.274    0.632    1.640     TRUE 0.916 1.000
## a[25,2]     -0.254  0.430   -1.146   -0.242    0.560     TRUE 0.724 1.001
## a[26,2]     -0.521  0.551   -1.693   -0.497    0.526     TRUE 0.837 1.000
## a[27,2]     -0.135  0.456   -1.037   -0.124    0.762     TRUE 0.609 1.002
## a[28,2]     -0.045  0.457   -0.944   -0.046    0.863     TRUE 0.543 1.000
## a[29,2]     -0.736  0.338   -1.416   -0.722   -0.092    FALSE 0.989 1.000
## a[30,2]     -0.632  0.598   -1.877   -0.624    0.520     TRUE 0.875 1.000
## a[31,2]     -0.781  0.631   -2.127   -0.748    0.335     TRUE 0.904 1.000
## a[32,2]      0.011  0.694   -1.356   -0.009    1.447     TRUE 0.496 1.000
## a[33,2]     -0.752  0.538   -1.842   -0.730    0.245     TRUE 0.928 1.001
## a[34,2]      0.237  0.682   -1.074    0.227    1.594     TRUE 0.635 1.000
## a[35,2]     -0.741  0.305   -1.357   -0.727   -0.178    FALSE 0.994 1.000
## a[36,2]      0.029  0.636   -1.240    0.043    1.289     TRUE 0.527 1.000
## a[37,2]      0.797  0.456   -0.073    0.782    1.725     TRUE 0.964 1.003
## a[38,2]      0.432  0.550   -0.563    0.420    1.565     TRUE 0.785 1.001
## a[39,2]     -0.159  0.510   -1.163   -0.168    0.850     TRUE 0.621 1.000
## a[40,2]      0.519  0.666   -0.764    0.501    1.890     TRUE 0.793 1.000
## a[1,3]       0.005  0.452   -0.938    0.031    0.847     TRUE 0.529 1.004
## a[2,3]       0.161  0.399   -0.616    0.162    0.963     TRUE 0.663 1.002
## a[3,3]       0.107  0.433   -0.727    0.100    0.991     TRUE 0.597 1.001
## a[4,3]       0.392  0.471   -0.437    0.354    1.463     TRUE 0.805 1.001
## a[5,3]       0.333  0.473   -0.523    0.300    1.367     TRUE 0.768 1.003
## a[6,3]      -0.040  0.433   -0.937   -0.028    0.772     TRUE 0.530 1.002
## a[7,3]      -0.034  0.459   -1.006   -0.014    0.865     TRUE 0.513 1.003
```

```
## a[8,3]      -0.018  0.446  -0.939   0.001   0.848    TRUE 0.499 1.001
## a[9,3]      -0.093  0.389  -0.915  -0.077   0.628    TRUE 0.583 1.001
## a[10,3]      0.220  0.470  -0.662   0.204   1.191    TRUE 0.689 1.002
## a[11,3]      0.340  0.475  -0.526   0.302   1.400    TRUE 0.773 1.005
## a[12,3]     -0.164  0.401  -1.020  -0.135   0.548    TRUE 0.641 1.002
## a[13,3]     -0.083  0.419  -0.969  -0.062   0.679    TRUE 0.563 1.001
## a[14,3]     -0.025  0.425  -0.884  -0.012   0.759    TRUE 0.512 1.001
## a[15,3]      0.023  0.414  -0.803   0.026   0.868    TRUE 0.527 1.000
## a[16,3]      0.175  0.395  -0.551   0.163   1.001    TRUE 0.670 1.003
## a[17,3]     -0.154  0.453  -1.140  -0.122   0.661    TRUE 0.622 1.000
## a[18,3]     -0.004  0.448  -0.993   0.022   0.830    TRUE 0.477 1.001
## a[19,3]      0.113  0.465  -0.815   0.103   1.069    TRUE 0.601 1.002
## a[20,3]      0.017  0.439  -0.896   0.029   0.885    TRUE 0.527 1.003
## a[21,3]      0.115  0.464  -0.819   0.114   1.051    TRUE 0.610 1.004
## a[22,3]      0.341  0.461  -0.475   0.312   1.364    TRUE 0.776 1.001
## a[23,3]      0.011  0.427  -0.823   0.020   0.827    TRUE 0.523 1.000
## a[24,3]     -0.122  0.447  -1.094  -0.086   0.669    TRUE 0.594 1.002
## a[25,3]     -0.192  0.422  -1.113  -0.171   0.597    TRUE 0.662 1.003
## a[26,3]     -0.045  0.418  -0.934  -0.026   0.742    TRUE 0.526 1.002
## a[27,3]     -0.074  0.418  -0.950  -0.060   0.743    TRUE 0.564 1.002
## a[28,3]      0.351  0.458  -0.466   0.316   1.373    TRUE 0.787 1.003
## a[29,3]     -0.050  0.379  -0.856  -0.039   0.658    TRUE 0.543 1.000
## a[30,3]      0.108  0.467  -0.814   0.103   1.074    TRUE 0.592 1.001
## a[31,3]      0.040  0.456  -0.903   0.053   0.909    TRUE 0.553 1.000
## a[32,3]      0.042  0.455  -0.843   0.037   0.949    TRUE 0.537 1.005
## a[33,3]      0.061  0.440  -0.828   0.059   0.946    TRUE 0.558 1.002
## a[34,3]     -0.079  0.473  -1.152  -0.050   0.778    TRUE 0.552 1.002
## a[35,3]      0.441  0.398  -0.258   0.418   1.331    TRUE 0.875 1.002
## a[36,3]      0.033  0.459  -0.903   0.037   0.969    TRUE 0.535 1.003
## a[37,3]      0.143  0.416  -0.688   0.143   1.016    TRUE 0.645 1.002
## a[38,3]     -0.096  0.452  -1.115  -0.066   0.706    TRUE 0.559 1.002
## a[39,3]      0.249  0.442  -0.555   0.227   1.189    TRUE 0.715 1.006
## a[40,3]      0.080  0.465  -0.879   0.086   0.979    TRUE 0.581 1.002
## b[1,1]      -1.042  0.775  -2.509  -1.086   0.552    TRUE 0.913 1.003
## b[2,1]      -0.261  0.401  -1.029  -0.271   0.571    TRUE 0.754 1.003
## b[3,1]      -1.204  0.690  -2.469  -1.234   0.239    TRUE 0.955 1.000
## b[4,1]      -1.044  0.496  -2.039  -1.056  -0.019   FALSE 0.976 1.001
## b[5,1]      -1.302  0.739  -2.692  -1.344   0.305    TRUE 0.953 1.001
## b[6,1]      -1.245  0.494  -2.224  -1.227  -0.311   FALSE 0.993 1.003
## b[7,1]      -1.183  0.679  -2.483  -1.213   0.209    TRUE 0.956 1.008
## b[8,1]      -0.996  0.760  -2.400  -1.025   0.542    TRUE 0.904 1.014
## b[9,1]      -0.170  0.465  -1.059  -0.175   0.780    TRUE 0.657 1.000
## b[10,1]     -1.500  0.758  -2.957  -1.504   0.055    TRUE 0.970 1.001
## b[11,1]     -1.377  0.739  -2.821  -1.380   0.069    TRUE 0.967 1.008
## b[12,1]     -0.001  0.529  -0.937  -0.047   1.163    TRUE 0.540 1.003
## b[13,1]      0.067  0.688  -1.133   0.001   1.491    TRUE 0.500 1.001
## b[14,1]     -0.583  0.699  -1.801  -0.646   0.978    TRUE 0.820 1.001
## b[15,1]     -0.777  0.531  -1.773  -0.796   0.352    TRUE 0.928 1.002
## b[16,1]     -0.340  0.576  -1.347  -0.392   0.972    TRUE 0.753 1.003
## b[17,1]     -1.078  0.658  -2.256  -1.129   0.351    TRUE 0.944 1.000
## b[18,1]     -1.276  0.642  -2.574  -1.270  -0.061   FALSE 0.981 1.005
## b[19,1]     -1.620  0.883  -3.391  -1.619   0.116    TRUE 0.967 1.016
## b[20,1]     -1.329  0.639  -2.605  -1.334  -0.092   FALSE 0.982 1.004
## b[21,1]     -1.685  0.862  -3.390  -1.684   0.009    TRUE 0.974 1.003
## b[22,1]     -1.481  0.659  -2.797  -1.461  -0.233   FALSE 0.986 1.000
## b[23,1]     -0.835  0.649  -2.084  -0.840   0.477    TRUE 0.912 1.000
## b[24,1]     -0.841  0.706  -2.217  -0.856   0.618    TRUE 0.886 1.004
```

```
## b[25,1]    -0.549  0.496   -1.474   -0.571    0.473     TRUE 0.868 1.002
## b[26,1]    -0.816  0.605   -1.904   -0.844    0.493     TRUE 0.906 1.002
## b[27,1]    -0.389  0.641   -1.539   -0.431    0.998     TRUE 0.747 1.000
## b[28,1]    -0.776  0.656   -1.916   -0.828    0.695     TRUE 0.886 1.013
## b[29,1]     0.259  0.479   -0.606    0.230    1.247     TRUE 0.698 1.003
## b[30,1]    -1.782  0.835   -3.370   -1.807   -0.107    FALSE 0.979 1.010
## b[31,1]    -1.376  0.751   -2.824   -1.375    0.130     TRUE 0.967 1.002
## b[32,1]    -1.491  0.778   -2.886   -1.517    0.150     TRUE 0.967 1.015
## b[33,1]    -1.194  0.447   -2.103   -1.191   -0.326    FALSE 0.996 1.000
## b[34,1]    -1.465  0.855   -3.120   -1.474    0.196     TRUE 0.957 1.007
## b[35,1]     1.212  0.530    0.326    1.180    2.354    FALSE 0.998 1.005
## b[36,1]    -1.537  0.749   -3.037   -1.534   -0.035    FALSE 0.977 1.003
## b[37,1]    -0.618  0.598   -1.723   -0.630    0.668     TRUE 0.863 1.005
## b[38,1]    -0.627  0.732   -1.922   -0.664    0.937     TRUE 0.820 1.002
## b[39,1]    -1.042  0.695   -2.350   -1.071    0.337     TRUE 0.932 1.001
## b[40,1]    -1.546  0.888   -3.276   -1.540    0.170     TRUE 0.961 1.014
## b[1,2]      1.033  0.764   -0.314    0.964    2.741     TRUE 0.931 1.004
## b[2,2]      0.490  0.386   -0.236    0.483    1.291     TRUE 0.903 1.001
## b[3,2]      1.053  0.639   -0.064    0.998    2.465     TRUE 0.966 1.007
## b[4,2]      0.186  0.463   -0.720    0.173    1.120     TRUE 0.659 1.000
## b[5,2]     -0.629  0.729   -2.152   -0.600    0.698     TRUE 0.810 1.002
## b[6,2]      0.748  0.476   -0.137    0.725    1.736     TRUE 0.954 1.000
## b[7,2]     -0.179  0.706   -1.613   -0.150    1.154     TRUE 0.585 1.003
## b[8,2]      1.085  0.773   -0.240    1.019    2.780     TRUE 0.939 1.001
## b[9,2]      1.240  0.533    0.310    1.208    2.370    FALSE 0.997 1.001
## b[10,2]     0.407  0.714   -0.876    0.386    1.955     TRUE 0.718 1.003
## b[11,2]     1.617  0.795    0.290    1.534    3.344    FALSE 0.993 1.004
## b[12,2]     0.556  0.522   -0.357    0.523    1.684     TRUE 0.865 1.000
## b[13,2]     0.425  0.548   -0.619    0.399    1.590     TRUE 0.796 1.000
## b[14,2]    -0.364  0.720   -1.847   -0.349    1.088     TRUE 0.697 1.002
## b[15,2]     1.435  0.761    0.222    1.326    3.170    FALSE 0.991 1.001
## b[16,2]    -0.221  0.662   -1.492   -0.220    1.094     TRUE 0.651 1.010
## b[17,2]     0.101  0.676   -1.134    0.074    1.600     TRUE 0.551 1.009
## b[18,2]     1.543  0.653    0.398    1.490    2.998    FALSE 0.999 1.001
## b[19,2]     0.871  0.841   -0.637    0.801    2.721     TRUE 0.867 1.004
## b[20,2]     1.316  0.875    0.072    1.150    3.548    FALSE 0.982 1.028
## b[21,2]     1.051  0.805   -0.302    0.971    2.978     TRUE 0.928 1.005
## b[22,2]     1.749  0.720    0.604    1.649    3.491    FALSE 0.999 1.005
## b[23,2]     1.775  0.754    0.558    1.719    3.466    FALSE 0.998 1.001
## b[24,2]     1.144  0.596    0.105    1.095    2.479    FALSE 0.985 1.004
## b[25,2]     1.305  0.529    0.375    1.266    2.443    FALSE 0.997 1.001
## b[26,2]     0.415  0.529   -0.540    0.381    1.516     TRUE 0.789 1.001
## b[27,2]     1.007  0.681   -0.161    0.958    2.518     TRUE 0.951 1.000
## b[28,2]    -0.220  0.636   -1.521   -0.215    1.029     TRUE 0.638 1.001
## b[29,2]     1.565  0.588    0.544    1.516    2.858    FALSE 1.000 1.001
## b[30,2]     0.087  0.708   -1.396    0.112    1.483     TRUE 0.565 1.001
## b[31,2]     1.142  0.741   -0.056    1.040    2.884     TRUE 0.967 1.012
## b[32,2]     0.232  0.744   -1.163    0.208    1.813     TRUE 0.626 1.003
## b[33,2]     0.829  0.522   -0.047    0.767    2.019     TRUE 0.966 1.002
## b[34,2]     0.459  0.869   -1.119    0.387    2.399     TRUE 0.700 1.005
## b[35,2]     0.969  0.537    0.036    0.935    2.144    FALSE 0.980 1.001
## b[36,2]    -0.308  0.687   -1.804   -0.270    0.941     TRUE 0.670 1.002
## b[37,2]     1.726  0.724    0.511    1.658    3.326    FALSE 0.999 1.004
## b[38,2]     0.780  0.646   -0.342    0.714    2.209     TRUE 0.909 1.004
## b[39,2]     1.793  0.818    0.454    1.704    3.665    FALSE 0.998 1.000
## b[40,2]     0.353  0.851   -1.252    0.333    2.132     TRUE 0.673 1.004
## deviance 1200.293 47.134 1112.661 1199.177 1297.446    FALSE 1.000 1.004
```

```
##           n.eff
## a[1,1]     290
## a[2,1]    2148
## a[3,1]    3000
## a[4,1]    3000
## a[5,1]    3000
## a[6,1]    1203
## a[7,1]     203
## a[8,1]     283
## a[9,1]     987
## a[10,1]   3000
## a[11,1]    197
## a[12,1]   1573
## a[13,1]    914
## a[14,1]   3000
## a[15,1]   3000
## a[16,1]   3000
## a[17,1]   1454
## a[18,1]    725
## a[19,1]    143
## a[20,1]    898
## a[21,1]    347
## a[22,1]   3000
## a[23,1]   1934
## a[24,1]   3000
## a[25,1]   3000
## a[26,1]   2910
## a[27,1]   1248
## a[28,1]    774
## a[29,1]   1382
## a[30,1]    405
## a[31,1]    479
## a[32,1]    341
## a[33,1]    989
## a[34,1]    231
## a[35,1]   1629
## a[36,1]   3000
## a[37,1]   1889
## a[38,1]    605
## a[39,1]   1605
## a[40,1]    195
## a[1,2]    3000
## a[2,2]    1994
## a[3,2]    2279
## a[4,2]    3000
## a[5,2]    3000
## a[6,2]    2147
## a[7,2]    2460
## a[8,2]    3000
## a[9,2]    3000
## a[10,2]   3000
## a[11,2]   2766
## a[12,2]   3000
## a[13,2]   1329
## a[14,2]   3000
## a[15,2]   2205
```

```
## a[16,2]   2685
## a[17,2]   2166
## a[18,2]   3000
## a[19,2]    532
## a[20,2]   1564
## a[21,2]   3000
## a[22,2]   3000
## a[23,2]   3000
## a[24,2]   2300
## a[25,2]   1609
## a[26,2]   2641
## a[27,2]   1404
## a[28,2]   3000
## a[29,2]   3000
## a[30,2]   3000
## a[31,2]   3000
## a[32,2]   3000
## a[33,2]   1543
## a[34,2]   3000
## a[35,2]   3000
## a[36,2]   3000
## a[37,2]    594
## a[38,2]   1576
## a[39,2]   3000
## a[40,2]   2738
## a[1,3]     984
## a[2,3]    1382
## a[3,3]    1683
## a[4,3]    2546
## a[5,3]    1023
## a[6,3]    1278
## a[7,3]    2376
## a[8,3]    3000
## a[9,3]    1088
## a[10,3]    913
## a[11,3]    420
## a[12,3]   1681
## a[13,3]   3000
## a[14,3]   3000
## a[15,3]   3000
## a[16,3]    870
## a[17,3]   3000
## a[18,3]   3000
## a[19,3]   1291
## a[20,3]   1130
## a[21,3]    886
## a[22,3]   1715
## a[23,3]   3000
## a[24,3]    894
## a[25,3]   2357
## a[26,3]    942
## a[27,3]   2529
## a[28,3]   1160
## a[29,3]   2655
## a[30,3]   3000
## a[31,3]   3000
## a[32,3]   1405
```

```
## a[33,3]    765
## a[34,3]   1046
## a[35,3]   1293
## a[36,3]    706
## a[37,3]   1814
## a[38,3]   2173
## a[39,3]    405
## a[40,3]   1086
## b[1,1]     621
## b[2,1]     696
## b[3,1]    3000
## b[4,1]    1135
## b[5,1]    3000
## b[6,1]     667
## b[7,1]     260
## b[8,1]     168
## b[9,1]    2214
## b[10,1]   3000
## b[11,1]    261
## b[12,1]    709
## b[13,1]   1627
## b[14,1]   1630
## b[15,1]   3000
## b[16,1]   3000
## b[17,1]   3000
## b[18,1]    388
## b[19,1]    137
## b[20,1]    443
## b[21,1]   1662
## b[22,1]   3000
## b[23,1]   3000
## b[24,1]   2480
## b[25,1]   1205
## b[26,1]   1036
## b[27,1]   3000
## b[28,1]    185
## b[29,1]   1182
## b[30,1]    231
## b[31,1]    976
## b[32,1]    154
## b[33,1]   3000
## b[34,1]    303
## b[35,1]    685
## b[36,1]    643
## b[37,1]    530
## b[38,1]   1159
## b[39,1]   3000
## b[40,1]    164
## b[1,2]    1511
## b[2,2]    1886
## b[3,2]     373
## b[4,2]    3000
## b[5,2]    1807
## b[6,2]    3000
## b[7,2]    1329
## b[8,2]    3000
## b[9,2]    2665
```

```
## b[10,2]    911
## b[11,2]    609
## b[12,2]   3000
## b[13,2]   3000
## b[14,2]    778
## b[15,2]   1106
## b[16,2]    409
## b[17,2]    487
## b[18,2]   2706
## b[19,2]    834
## b[20,2]    248
## b[21,2]    724
## b[22,2]   1238
## b[23,2]   3000
## b[24,2]    516
## b[25,2]   3000
## b[26,2]   2110
## b[27,2]   2501
## b[28,2]   3000
## b[29,2]   3000
## b[30,2]   2063
## b[31,2]    316
## b[32,2]    838
## b[33,2]   3000
## b[34,2]   3000
## b[35,2]   3000
## b[36,2]   3000
## b[37,2]   2015
## b[38,2]    555
## b[39,2]   3000
## b[40,2]   1021
## deviance   479
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 1106.9 and DIC = 2307.203
## DIC is an estimate of expected predictive error (lower is better).
```

What now with all these numbers and letters?

So you managed to get your model to run (hopefully) and converge.