**Bayesian Statistics Practical**

**CEH R Conference 23rd-24th November 2016**

**Lindsay Banin**

**Pre-requisites**

* Check you are running a recent version of R
* Install ‘rjags’ package in R
* Go to <http://sourceforge.net/projects/mcmc-jags/files/JAGS/>. Click on Windows and then the JAGS executable (exe file).
* Running multiple versions of R on your computer may cause problems!
* Install ‘coda’ and ‘lattice’ packages in R

**Exercise 1: Modelling light limitation of Hemlock tree growth**

*Files*

Data: Hemlock-light-data.csv

R command: Tree light example.R

JAGS command: jags\_light\_example.R

*Introduction*

The example dataset and analysis were provided by Maria Uriarte and adapted and annotated for this session.

The relationship between tree growth rate and light tends to be non-linear, approaching an asymptote under high light conditions. Here, we model this simple curve using a Bayesian approach, where our response (y) is observed growth rate and our only predictor variable is light (L).

We use the below (Fig. 1) common function to describe the relationship between tree growth and light. Note, in the accompanying R code, greek letters alpha and gamma are given by letters a and b respectively. First, load the data and explore it by plotting and applying a non-linear least squares regression. Note the results of your NLS analysis. You could play with the starting values to examine whether these are influential in the findings. You could also analyse the data using maximum likelihood, for instance using Ben Bolker’s bblme package.

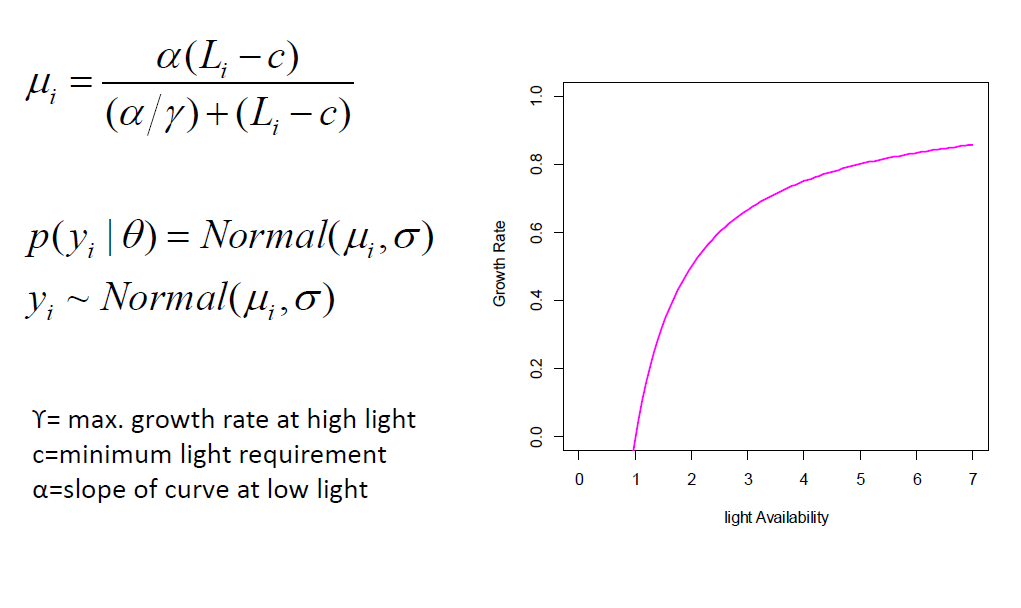


Figure 1. Relationship between light availability and tree growth rate.

A directed acyclic graph (or DAG) can be used to visualise the dependencies between components of the model. Y is determined by data L and parameters a, b, c (from within the non-linear function specified in Fig 1.) and the variance parameter, sigma. Each parameter will have a prior distribution (with hyperparameters). You could conceive the priors as the range of expected values (with associated probabilities) for the parameter. See other examples and a fuller description of DAGs in Ogle & Barber (2012).

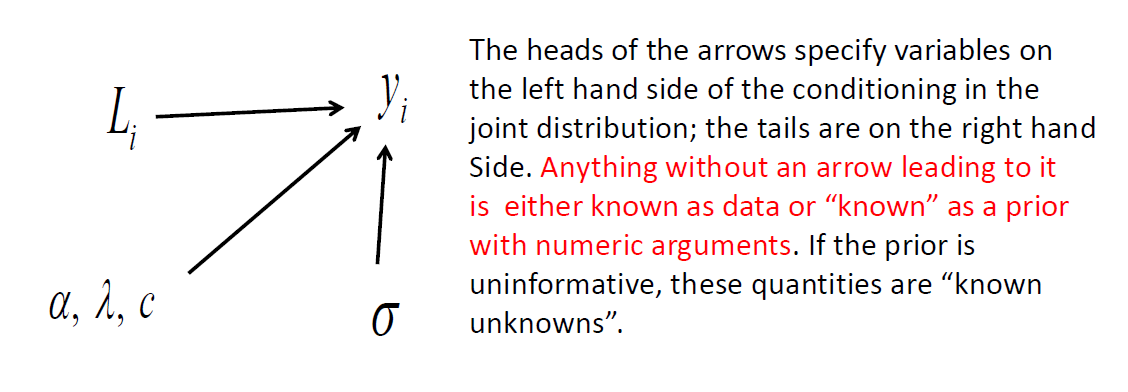


Figure 2: DAG for light-growth model

*The JAGS code: likelihood and priors*

Examine the JAGS code. All Bayesian models in JAGS contain:

1. Code for the deterministic model
2. Code for the likelihood(s)
3. Code for the priors

They appear in this order in the example code, but in contrast to R command, the code does not need to appear in a specific order. Note the code uses the precision parameter tau, which is 1/σ2.

The Likelihood

When specifying an *individual* likelihood, we ask, what is the probability (density) that we would obtain this data point conditional on the value of the parameters of interest? The *total* likelihood is the product of individual likelihoods. The **for** loop in the JAGS code specifies the elements in the product of the likelihoods.

From this we can create a likelihood profile (the likelihood of the data, given the parameter values) (Fig. 3). NB this is not a distribution; the area under the curve does not integrate to 1. Also, the absolute value of the y axis does not hold meaning – we only care about it in relative terms (e.g. comparing model fit).

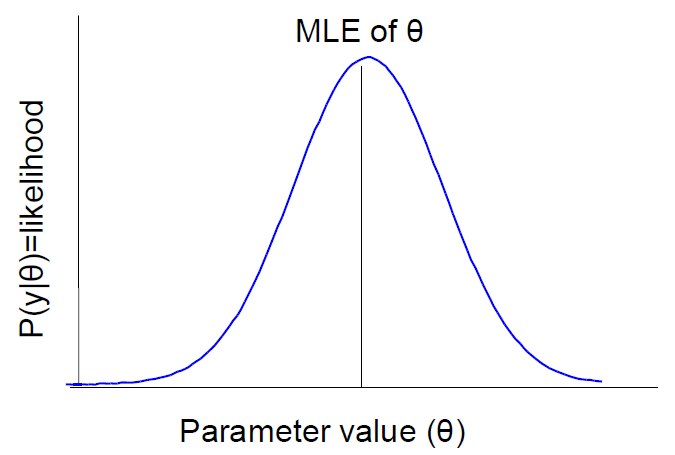


Figure 3. Likelihood profile

The Priors

A quick note on conjugacy – when the prior distribution and the posterior distribution are of the same family, they are said to be conjugate. Look up tables can be highly useful is identifying conjugate distributions (e.g. <https://en.wikipedia.org/wiki/Conjugate_prior#Table_of_conjugate_distributions>). See Ogle & Barber (2012) for a more detailed discussion.

The support for the variance and parameters a and b are that they should be positive, real numbers. A good candidate distribution to meet these conditions is the gamma distribution; the gamma distribution is also a conjugate to the normal distribution (the distribution of our likelihood). Parameter c could arguably take on positive or negative numbers – here we use a uniform distribution as the prior for c (i.e. equal probability for all values within bounds).

Plot the priors. Later you could update the JAGS model to change the specification of the priors – the shape and rate parameters - how does this affect your posterior distributions of the parameters?

*Implementing the model in R*

In R, we must set the initial conditions for the MCMC chain(s). It is efficient to get one chain running well before adding computing time of additional chains. In this simple example, however, we launch directly into producing three chains. The initial conditions are specified as a list of lists. As we will run 3 chains, we require 3 starting values for each parameter to be estimated. Recall the DAG (Fig. 2) – each component which is not data requires a start value. JAGS will choose start values if you do not specify them, but this can be risky, especially if priors are vague.

Next, we specify the data to be used by the JAGS model. Notice that we name the R data variables x and y to match the JAGS code. We also specify the form the data must take (numeric).

To actually run the JAGS model, first we specify the number of iterations in the chain to use for adaptation, for burnin and for the final number of iterations in the chain. Adaptation is the number of iterations used for JAGS to choose an appropriate sampler, to ensure optimum mixing of the chain (recall that we want to chain to look like a hairy caterpillar!). Burnin refers to early samples, which may be far from the parameter value, which will be discarded (in our case, 1000). The number of iterations is the number of samples used to form the posterior distribution.

The call jags.model makes use of the JAGS code file, the data and conditions we have specified. Note that your JAGS file should be in your same working directory, otherwise you need to specify the full file path. The number of chains (n.chains) is determined by the length of the initial values – in our case, 3.

*Coda and jags objects*

Coda.samples traces the chains and stores them as an MCMC list. We list the variable names we wish to trace and save; these are the variables for which we want to estimate the posterior distribution (i.e. our parameters). Chains can be thinned so that we only store every nth element in the chain. This can be helpful to reduce the size of the files to store and analyse, particularly if long chains are required for convergence. Here we create an object (zm), from which we can produce a summary.

We also produce a JAGS object using call jag.samples. Coda.samples tend to be useful for producing summary statistics and jags.samples for plotting. Investigate the structure of the two objects produced, and visit the R documentation for each.

Plot the trace and density plots using the zm object. Notice that you have separate plots for each parameter and three coloured lines – one for each MCMC chain. These are more visible if you thin your data (e.g. thin = 10) in your coda.samples.

*Convergence*

Visit the R documentation for each of the diagnostic tests to examine what each of them do. What do you determine about the chains for each parameter – are you happy with the mixing and convergence?

Play with the values for n.adapt, n.update and n.iter, and the ‘thin’ argument in your coda.samples and jags.samples calls; re-run the code up to plot(zm, ask = TRUE) to investigate how these alter your chains and samples in your posterior distributions.

*Drawing inference*

In the summary of the zm object, identify the estimated means for each parameter. How do they compare with NLS estimates at the beginning of the exercise? (You may notice that in the JAGS model, the deterministic function has a different sign for parameter c).

We can show uncertainty around a point estimate by defining a Bayesian credible interval, with upper and lower bounds (i.e. the probability that the true value of the random parameter falls within the interval). This can be done by approximating the interval bounds using quantiles of the MCMC samples (NB they may not always be symmetric!). Use the code to identify and plot the 95% credible intervals using this method.

**Exercise 2: Extending the model (open exercise)**

This is a very simple model. It is worth exploring how you might extend this model.

1. How would you define the model if (i) you had data from more than one site, (ii) you had another covariate (e.g. soil nutrient availability, (iii) you had data from another species.
2. Draw a DAG for a chosen more complex model, using information in Ogle & Barber (2012) to help guide you. What additional parameters would be included in your posterior? What priors would each parameter require?
3. Amend the JAGS model to reflect these alterations.