Introduction to Bayesian Statistics

Introduction to WinBugs

**Running a model for a single Proportion in WinBugs**

We will now write code to model a very simple binomial experiment. Go to file and select New. This will open a text window that will contain the commands we will discuss below. All programs (models) in WinBugs start with the following lines:

model

{

This means that you are about to write something for Winbugs to run. The “{“ indicates a set of commands for WinBUGS to process. There should always be as many “{“ as there will be “}” as a open set must always be closed.

The model statement can be viewed as having two parts. The first is the likelihood that tells WinBUGS about the distribution of the data. The second part specifies the prior(s) as well as parameter values. We will deal with each of these in turn.

Likelihood

For the example we have been using, females with cancer, the data consists of 24 independent observations, 7 of which are 1s (i.e. 7 females in the data),, and the rest are 0s (i.e. 17 males in the data). Visually the data look like

|  |  |
| --- | --- |
| Cancer subject | x (1=Female, 0=Male) |
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
|  |  |
| 23 | 0 |
| 24 | 1 |

Each observation can be thought of as being generated from a Bernoulli distribution with some probability of “success” which in this case is the subject is a female. Our goal is to estimate that probability of success using these data. Recall the discussion we had about how the likelihood is computed, i.e. the product of the pdfs of each individual observation. The way we communicate this to WinBUGS is with a for loop

for (i in 1:24)

{

x[i]~dbern(success.prob) #P(x|theta)

}

**NOTE**: The command “for” indicates that we want to start a loop over specific numerical values. The symbol ~ is used to indicate what distribution a variable follows. The symbol [] is used access an element of a vector. dbern is the name for Bernoulli distribution. # can be used to make comments which will not be read as a command.

The above loop will replace i with then number 1 the first time through the loop, 2 the 2nd, and so on until it evaluates the Bernoulli distribution for the 24 data points. In this way the commands tell WinBUGS that a Bernoulli distribution should be assigned to each of the 24 observations in the data set with a success probability that will come from the prior information below.

Prior

Our goal with this analysis was to obtain a Bayesian estimate (posterior distribution) of the probability a cancer subject is a female given the observed data, P(female|x). To do this we specify a prior belief about that probability such as

success.prob~dbeta(alpha,beta) #P(theta)

alpha<-15

beta<-15

NOTE: dbeta is used to indicate a beta distribution. <- is like and equal sign (=). So here, we are saying that alpha=15 and beta=15.

Putting these two parts together the model statement then looks like

model

{

#likelihood

for (i in 1:24)

{

x[i]~dbern(success.prob) #P(x|theta)

}

#Prior

success.prob~dbeta(alpha,beta) #P(theta)

alpha<-15 #parameter for prior

beta<-15 #parameter for prior

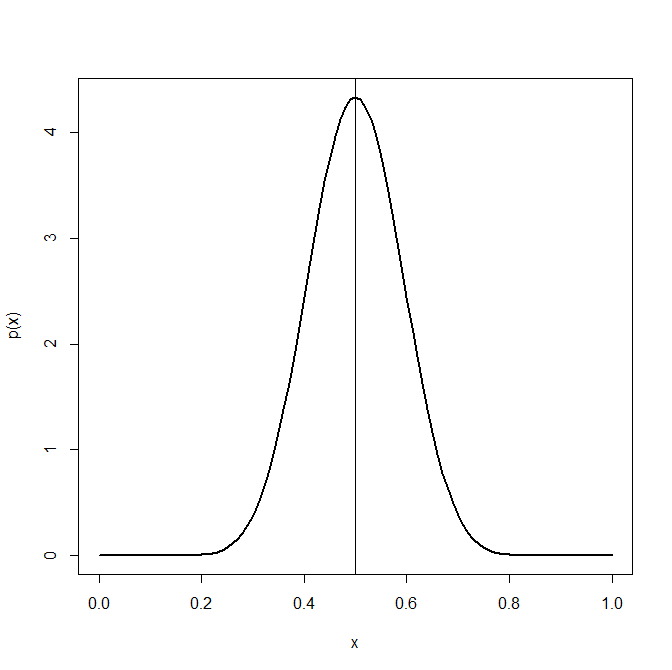
}

Next, we need to input our data. This can be done as follows

#Data

list(x=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))

Finally, to get the Markov chain going, we need to specify a place for it to start. This initial value reflects a point in the domain of the prior distribution so considering the shape of the prior may be helpful. In our case we are using a Beta(15,15) which will be symmetric with mode around 0.5 on the domain 0 to 1. In fact the exact shape of the distribution is given below.



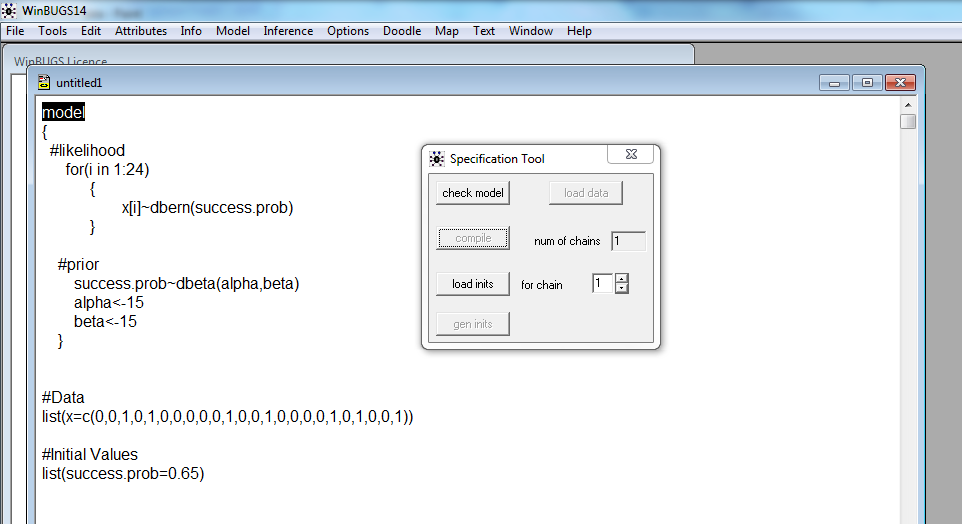
We can see that by starting the chain anywhere between say .3 and .7 will be in the part of the Beta(15,15) distribution with the most area, and, barring data that are very different from the prior, will likely to be in the part of the posterior that has the most area as well. This means the chain will start exploring the “meat” of the parameter space quickly. Starting the chain out in the tails, in this case below say 0.1 or above say 0.9 might take the chain a while to work its way into the center of the distribution and can cause trouble with the chain getting to the actual posterior we want to sample from. In fact, if it is too far in the tails, the chain can sometimes wander the wrong direction and never reach the area we want it to (more on this when we discuss MCMC diagnostics). Thus it is a good idea to pick an initial value near the center of the prior distribution (more on this and exceptions to it as well later). In fact another Gibbs sampling package called JAGS simply takes the mean of the prior as the initial value. WinBUGS can generate these for us but does not take the mean of the distribution, rather it randomly selects a value from the prior distribution, which, granted, is likely to be near the center of the distribution but does not preclude it from selecting out in the tails. This is much less of an issue when using strong priors and it much more of an issue when using vague priors. Consider a Beta (1,1) distribution in this case. What points are the most likely for that distribution? We assign an initial value as outlined below

#Initial values

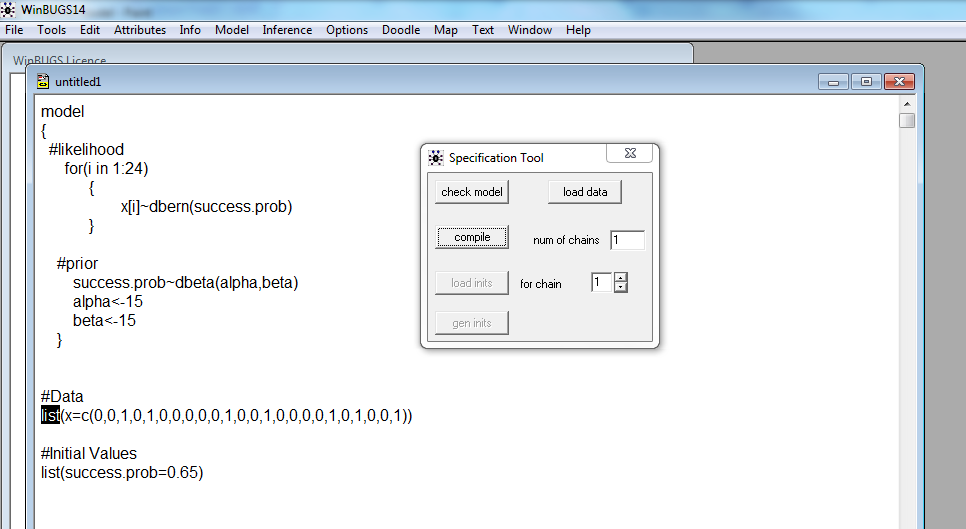
list(success.prop=0.65)

With all of this in place, we are ready to feed this information in to WinBUGS to start the Gibbs sampler.

From the Model menu, select “Specification…” and the Specification Tool will appear. All buttons will be greyed out at first. The first thing you will want to do is double click on “model” within the commands you have written. The ‘check model’ button will become available within the specification tool. Click on it to load your model statement.

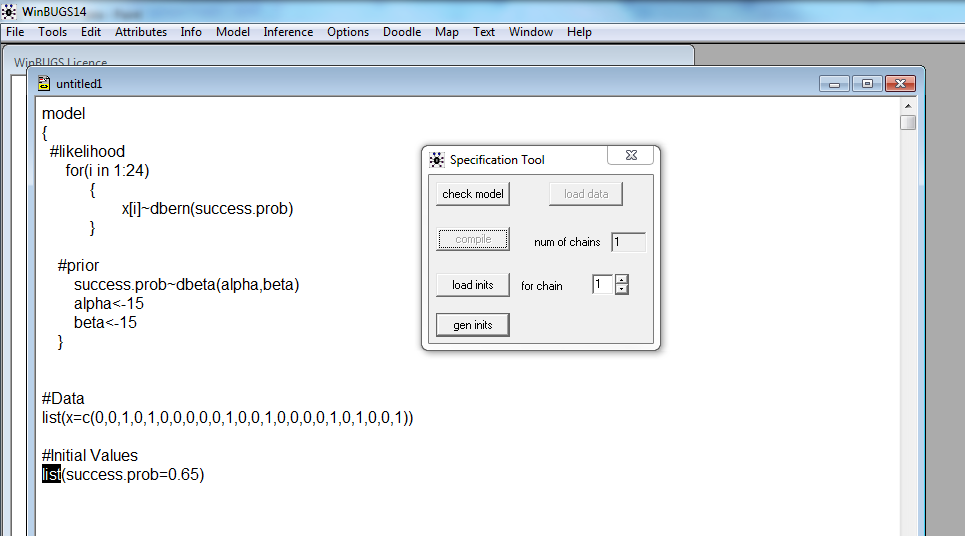


WinBUGs will check your model statement at that time and a message will appear at the bottom left of the window to tell you if your it has any syntactical errors (these error messages can be a little cryptic). If everything is good, you will get the message “model is syntactically correct”. Once the model has been checked and is ok, the button “load data” will not be greyed out anymore, meaning that you are able to load your data. To do this, select and double-click on the word “list” at the beginning of your data and select the “load data” button.

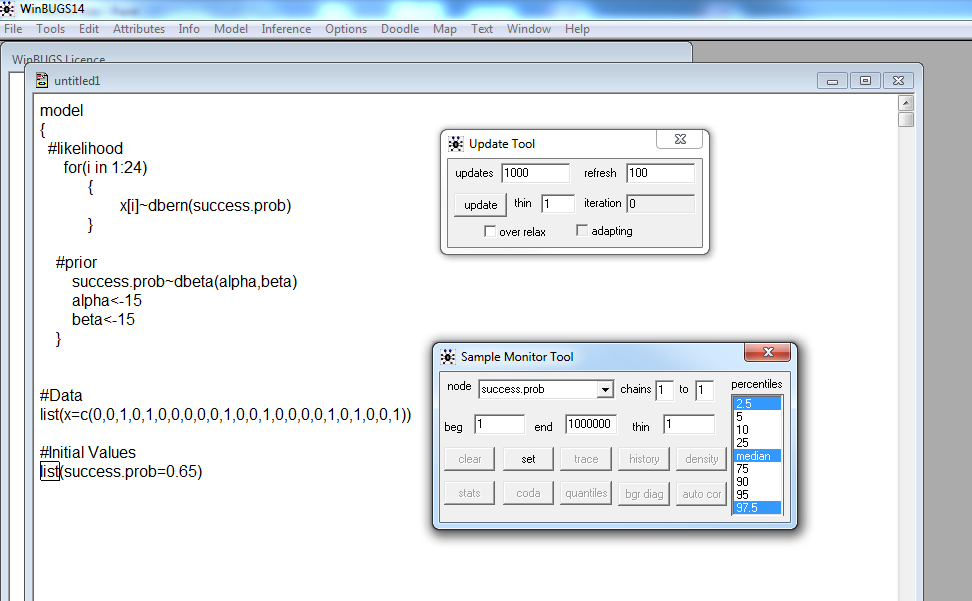


You will once more see a message appear at the bottom of the window to let you know if the data is ok. If so, you will have the message “data loaded”. Otherwise, you will soon find out all the possible error messages!

The next thing to do is to click on “compile” on the Specification Tool window. This will “check” for additional errors. For example, if distributions are well specified, if the number of observations correspond to your data etc. If all is well, you will get the message “model compiled” at the bottom of WinBugs window. If not..trouble starts…. We will certainly get some examples of crashes during this workshop… Once the model has been compiled, we can then upload the initial values. As with the data, we will select and double click on the word “list” for the initial values and then select “load inits” from the Specification Tool. The phrase “model is initialized” will appear at the bottom of the window if the initial value(s) were correctly specified.

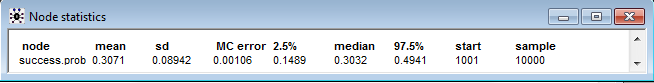


The model is then ready to run. To do so, select Model, Update in the menu bar. A new window called “Update Tool” will appear. This window asks you how many times you want to sample from the joint posterior distribution. Usually, we will run a few “burn-in” updates to simply get the model to stabilize a little. The updates are defaulted at 1000, but you can change this number. Click on update, you will see the number of iterations already run in the “iteration” box. Once this is done, it is time to tell WinBBUGS which variable you are actually interested in estimating. In our case, it is the posterior distribution of the prevalence of females, which we called success.prob. In the menu bar, select Inference, samples. In the node window, type success.prob and click on set. This tells WinBugs to “monitor” (record) the different values of success.prob which will be estimated from combining the prior and likelihood values, in other words, your posterior.

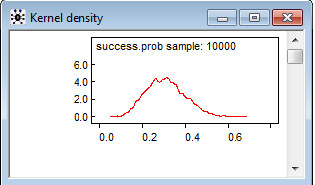


Once this is done, go back to the Update Tool and run 10,000 updates.

You are now ready to get your results. On the Sample Monitor Tool window, the box “stats” will give you the statistics for all variables (nodes) monitored. This will give you summary measures of the posterior distribution.



The button “density” gives you a visual of the posterior.



We will discuss some of the other buttons later that are useful for determining how well the MCMC process worked.

**BONUS**

There is a nice relationship between the Bernoulli and Binomial distributions which allows us a short cut for specifying the likelihood and data in WinBUGS. The relationship is this….If an experiment consists of independent Bernoulli trials (ie each outcome is binary) with the same probability of success on every trial, then the number of successes follows a Binomial distribution with parameters and . In statistical short-hand we would write this as…

Let , then .

Because the “for” loop must be used in other situations, we present it first as the standard, but should also point out that, in this case, the following would work as well.

model

{

#likelihood

y~dbin(success.prob,n)

#Prior

success.prob~dbeta(alpha,beta) #P(theta)

alpha<-15 #parameter for prior

beta<-15 #parameter for prior

}

#Data

list(n=24,y=7)

This is extremely useful when modeling a success probability in a univariate setting (not needing to include other covariates in the model) and the sample size is large.