runjags: An R Package Providing Interface Utilities, Distributed Computing Methods and Additional Distributions For MCMC Models in JAGS

Matthew J Denwood
University of Glasgow

Abstract

The **runjags** package provides a set of interface functions to facilitate running Markov chain Monte Carlo models in JAGS from within R. Automated calculation of appropriate convergence and sample length diagnostics, user-friendly access to commonly used graphical outputs and summary statistics, and parallelised methods of running JAGS are provided. The primary motivation is to provide automated methods of analysis of simulated data to facilitate model performance assessment and drop-k type cross-validation studies using high performance computing clusters provided by **snow**. A module extension for JAGS providing the Pareto family of distributions is also included within **runjags**. This vignette is taken from the publication for the **runjags** package (Denwood [In Review]). It outlines the primary functions of this package, and gives an illustration of a simulation study to assess the sensitivity of a gamma-Poisson (negative binomial) distribution to three different 'minimally informative' priors.

Keywords: MCMC, Bayesian, graphical models, interface utilities, JAGS, **BUGS**, R.

1. Introduction

Over the last decade, the increase in availability of computing power has led to a substantial increase in the availability and use of computationally intensive statistical methods; amongst the most widely adopted of these are Bayesian Markov chain Monte Carlo (MCMC) methods (Gilks, Richardson, and Spiegelhalter 1998). However, such methods have potential drawbacks if used inappropriately, including difficulties identifying convergence (Toft, Innocent, Gettinby, and Reid 2007; Brooks and Roberts 1998) and the potential for autocorrelation to decrease the effective sample size of the numerical integration process (Kass, Carlin, Gelman, and Neal 1998).

Although writing customised MCMC sampling algorithms is relatively straightforward, particularly using the Metropolis-Hastings algorithm (Hastings 1970), it has become more common practice to employ more general Bayesian MCMC fitting software such as the Bayesian analysis Using Gibbs Sampling (**BUGS**) software variants **WinBUGS** and **OpenBUGS** (Lunn, Thomas, Best, and Spiegelhalter 2000). One alternative known as Just Another Gibbs Sampler (JAGS) has more recently been made available by Plummer (2013a), and offers cross-platform support, a direct interface to R using **rjags** (Plummer 2013b), as well as being extendable with user specified modules written in C++ to add support for additional distri-

butions and random number generators (Wabersich and Vandekerckhove 2013). Both JAGS and **WinBUGS**/**OpenBUGS** use the BUGS syntax to allow arbitrary models to be more easily defined by the user, and provide sufficient flexibility to be used for the vast majority of biological statistical problems. This flexibility and ease of use makes using the BUGS language attractive and attainable for researchers who may be somewhat familiar with more traditional frequentist modelling techniques, but are not aware of the potential issues with MCMC analysis, hence the prominent warning that 'MCMC sampling can be dangerous' in the **WinBUGS** user manual (Lunn *et al.* 2000). One way to reduce some of this potential risk for inexperienced users would be to provide a wrapper for the model fitting software that analyses the model output for common problems, such as failure to converge, parameter auto-correlation and effective sample size, that may otherwise be missed by the end user.

Besides the potential problems associated with convergence and auto-correlation, one of the most commonly criticised aspects of Bayesian MCMC is the requirement for prior belief to be incorporated into the model. This is of course also a potentially huge advantage of Bayesian methods, and many methods exist for the elicitation of prior distributions to improve inference (Garthwaite, Kadane, and O'Hagan 2005), but the perceived lack of objectivity of informative priors leads some authors to advocate minimally informative priors (Kass and Wasserman 1996). There are a number of different recommendations for an appropriate choice of prior distribution in various different circumstances, for example the half-Cauchy distribution has been recommended as a reasonable choice for standard deviation parameters within hierarchical models (Gelman 2006; Polson and Scott 2011), and DuMouchel (1994) gives an argument for the use of $\pi(\tau) = \frac{s_0}{(s_0+\tau)^2}$ as a prior for a variance parameter τ in meta-analysis models. However, there are frequently several minimally informative priors that could be equally justifiable in a given situation, and choice between these is known to affect the shape of the posterior (Lele and Dennis 2009), particularly when the information in the data is relatively sparse. For example, Tuyl, Gerlach, and Mengersen (2008) reviewed the options for specifying a prior for Binomial events, and found that the Jeffreys prior in this situation performed poorly. The choice of prior distribution can also adversely affect model convergence and identifiability; particularly the `Gamma(0.001,0.001)` prior, which is often used for precision parameters but is almost improper. In addition, the flexibility of MCMC techniques and often complex structure of the data can lead to situations where multiple syntactically different but conceptually equivalent models could be devised, which also has the potential to affect the inference made. Validation of model formulation and prior selection using simulated data is a pragmatic way of guarding against these problems, but is rarely performed perhaps due to the computational effort required, and the requirement to assess the convergence and effective sample size of repeated MCMC simulations. Although assessment of these factors should always be performed manually in real-world situations, a method of automating these procedures may be warranted for analysis of simulated data to assess model performance.

This paper describes the **runjags** package for R which can be used to automate MCMC fitting and summarisation procedures for JAGS models, particularly for a simulation study with an arbitrary number of replicate datasets, and also provides additional distributions to extend the core functionality of JAGS. An example of usage to assess the sensitivity of an over-dispersed count observation model to various minimally informative prior distributions is also provided. Some prior familiarity with the BUGS programming language and the underlying MCMC algorithms is assumed.

2. Package functions

2.1. Basic usage

The core functionality of the runjags package allows a model specified by the user to be run using a variety of methods to call JAGS. This model can be specified in an external text file, which is likely to be preferable for more complex model formulations, or as a character string within R, which eliminates the need for multiple text files. External text files containing data and initial value lists compatible with **WinBUGS** are also supported, and will be converted to the required JAGS format (more details are given in the read.winbugs help page).

The model formulation can also contain three special inline statements which are interpreted by the runjags package: `#data#`, which indicates that the comma separated variable names to the right of the statement are to be included in the simulation as data, and similarly `#monitor#` which indicates variable names to monitor, and `#inits#` which indicates variables for which initial values are to be provided. Variables specified by `#data#` and `#inits#` will be automatically retrieved from a named list provided using the 'datalist' and 'initlist' arguments, or failing that from the parent environment for the function call. For initial values, a list of length equal to the number of chains containing initial values to be used for each chain should be provided; if only a single set of initial values is provided a warning will be given. The `#data#` and `#inits#` variable names may also match a function returning an appropriate vector, in the case of initial values this function may accept a single argument indicating the chain that the initial values are to be used for. Multiple inline statements can be used, and will simply be combined.

There are several options to the 'run.jags' function including explicit specification of monitors, data and initial values, the required burn in period, sampling length and thinning interval, the summary statistics to calculate from the chains, the method to use for calling the JAGS executable, and options for 'modules' and 'factories' which allow a character vector of JAGS extension modules and factories to be loaded. A basic model run with a fixed burn in period (default 4000 iterations after 1000 adaptive iterations) and sampling period (default 10000 iterations) can be obtained as follows.

Specify a JAGS model as a character vector:

```
R> model <- "model {
+         for(i in 1 : N){ #data# N
+         Y[i] ~ dnorm(true.y[i], precision) #data# Y
+         true.y[i] <- (coef * X[i]) + int #data# X
+  }
+  coef ~ dunif(-1000,1000)
+  int ~ dunif(-1000,1000)
+  precision ~ dexp(1)
+  #inits# coef, int, precision, .RNG.seed, .RNG.name
+  #monitor# coef, int, precision
+  }"
```

Simulate the data:

```
R> set.seed(1)
```

```
R> X <- 1:100
R> Y <- rnorm(length(X), 2*X + 10, 1)
R> N <- length(X)
```

A function to return initial values (including RNG seeds) for each chain:

```
R> coef <- function(chain)
+  return( switch(chain, "1"= -10, "2"= 10) )
R> int <- function(chain)
+  return( switch(chain, "1"= -10, "2"= 10) )
R> precision <- function(chain)
+  return( switch(chain, "1"= 0.01, "2"= 100) )
R> .RNG.seed <- function(chain)
+  return( switch(chain, "1"= 1, "2"= 2) )
R> .RNG.name <- function(chain)
+  return( switch(chain, "1" = "base::Super-Duper",
+  "2" = "base::Wichmann-Hill") )
```

Run the simulation:

```
R> results <- run.jags(model, n.chains = 2, method="interruptible")

Calling the simulation...
Welcome to JAGS 3.4.0 on Mon Nov 18 21:37:47 2013
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 407
. Reading parameter file inits1.txt
. Reading parameter file inits2.txt
. Initializing model
. Adapting 1000
-------------------------------------------------| 1000
+++++++++++++++++++++++++++++++++++++++++++++++++ 100%
Adaptation successful
. Updating 4000
-------------------------------------------------| 4000
************************************************* 100%
. . . . Updating 10000
-------------------------------------------------| 10000
************************************************* 100%
. . . . Updating 0
. Deleting model
```

```
.
Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 3 variables....
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

The simple model shown (a linear model with intercept and single explanatory variable) is run using two chains, contains three variables that are monitored, and convergence is assessed using the Gelman-Rubin statistic (Gelman and Rubin 1992). The method for running the model shown here does so by calling an external JAGS process in the foreground, but a variety of methods are available: 'parallel' runs separate JAGS processes for parallel chains, 'background' and 'bgparallel' run the model in the background with control of the R interface returning to the user while it is running, 'snow' runs parallel chains in separate JAGS processes using a (possibly user-specified) distributed computing cluster, 'rjags' uses the **rjags** package to run JAGS, and 'rjparallel' runs parallel chains using separate **rjags** models using a distributed computing cluster. The parallel methods ('parallel', 'bgparallel' and 'rjparallel') should speed up computation of models with multiple chains on multi-core machines, and all issues pertaining to pseudo-random number generation are automatically handled by **runjags**: in this example the RNG type and seed is set for each chain, but if none is provided each chain will automatically be given a different random number generator in order to avoid possible issues with non-independence between chains. The method used for each model run can be controlled directly using the 'method' argument, or by changing the default using the 'runjags.options' function to permanently set an alternative method as the default. For example the following code will allow a possibly lengthy JAGS simulation to be run in the background using two processors to speed up the simulation:

```
R> info <- run.jags(model, n.chains = 2, method = 'bgparallel')

Starting the simulation in the background...
The JAGS process is now running in the background
```

This returns control of the terminal to the user, who can then carry on working in R while waiting for the simulation to complete, then retrieve the results once it has:

```
R> results <- results.jags(info)

Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 3 variables....
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

If the model did not converge or take sufficient samples from the initial JAGS call, the model can be extended (using a different method to call JAGS if desired) for a fixed additional number of samples with the 'extend.jags' function, with the initial results either being combined with

the new simulation or discarded. Alternatively, the simulation can be automatically extended using the 'autoextend.jags' function, which ensures that the Gelman-Rubin statistic for all monitored parameters meets the specified target to indicate convergence, and that the required number of samples is obtained. The automated assessment of convergence should always be verified manually for inference to be relied upon, but a fully automated analysis may be sufficient for simulated data. The initial simulation may also be called using 'autorun.jags' in place of 'run.jags' to fully automate control of the MCMC simulation run length from the start.

The output of these functions is an object of class 'runjags'. This class is associated with S3 methods for print, plot, as.mcmc and as.mcmc.list - all of which may take a 'vars' argument to specify a subset of monitored nodes (using partial matching). Further utility functions are available for combining multiple objects ('combine.mcmc') and for conversion to/from objects produced by the rjags package ('as.runjags' and 'as.jags'). The plot method is intended to be used for convergence diagnostics, and has further arguments for commonly used layout and plot type arguments, but is not intended to be used for producing more specific graphical output from converged MCMC chains for which plot methods associated with 'mcmc' or 'mcmc.list' objects are more appropriate. A typical examination of the output of a simulation (the default print method, and a trace plot output for variable names partially matching the letter 'c') could be obtained as follows:

```
R> results


JAGS model summary statistics from 20000 samples
(chains = 2; burnin = 5000):

          Lower95 Median Upper95   Mean        SD
coef       1.9935 1.9995   2.006 1.9996 0.0031642
int        9.7534  10.13   10.48 10.131   0.18577
precision 0.89024 1.2131  1.5533 1.2213   0.17145

                MCerr MC%ofSD SSeff     AC.10   psrf
coef      0.000076615     2.4  1706    0.1895 1.0003
int         0.0044427     2.4  1748   0.19662 1.0002
precision   0.0012484     0.7 18862 0.0053645      1

Total time taken: 11.9 seconds

R> plot(results, type = "trace", vars = "c", layout = c(2,1) )

Producing 2 plots for 2 variables to the active graphics device
(see ?runjagsclass for options to this S3 method)
```

The print method displays information on the median and 95% credible interval (CI) estimates, as well as the mean, standard deviation, Monte Carlo error, Monte Carlo error as a proportion of sample standard deviation, the effective sample size, autocorrelation at lag 10,

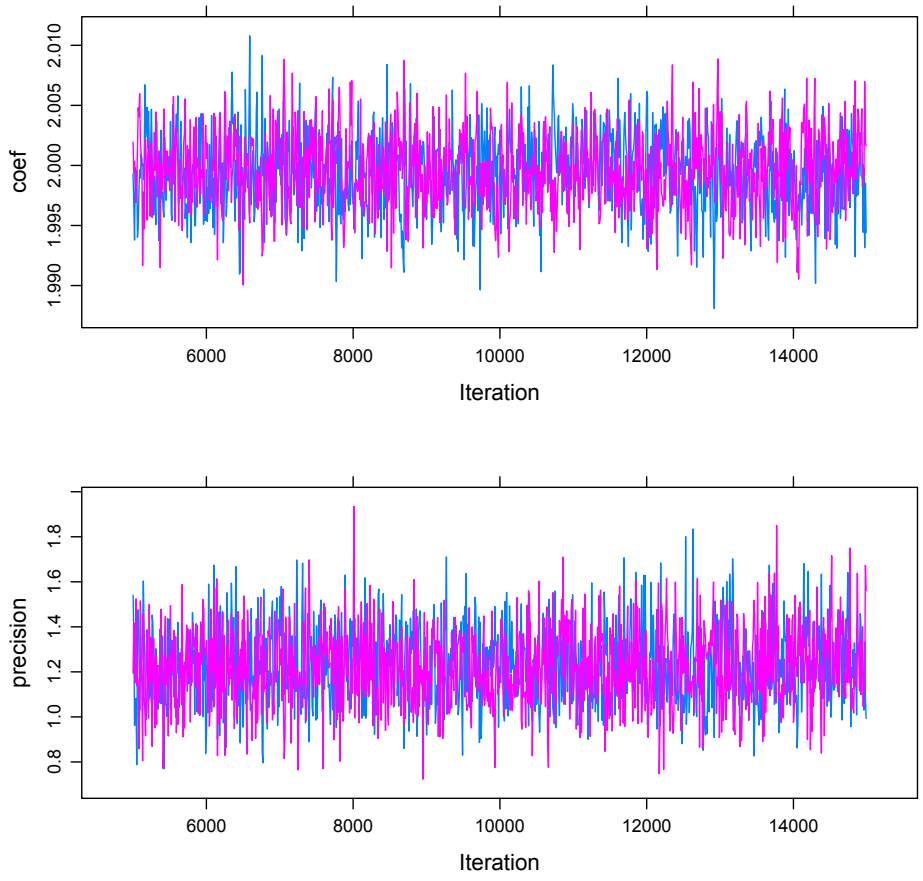

Figure 1: A traceplot displayed by the plot method for the runjags class, showing only parameters partially matched using the letter 'c'. Multiple chains are shown using different colours.

and potential scale reduction factor of the Gelman-Rubin statistic (Gelman and Rubin 1992; Brooks and Gelman 1998). The plot method displays trace, density and/or cross-correlation plots for the variables specified, using either a single graphics device or by opening multiple devices, or by saving the graphical output to a file for inspection at a later time. The **coda** package (Plummer, Best, Cowles, and Vines 2006) provides many of the underlying functions that calculate these statistics.

2.2. Simulation studies

The principle motivation behind development of the **runjags** package is to automate the analysis of simulated datasets for the purposes of model validation. While it is possible to repeatedly use the 'autorun.jags' function to analyse multiple datasets, a higher level 'run.jags.study' function is provided to automate much of this process. This function takes arguments specifying the number of datasets to analyse, the model to use, a function to produce data that will be provided to each simulation, and a named list of 'target' variables with true values representing parameters to be monitored and used to summarise the output of the simulation. Inline `#inits#` and `#monitor#` statements can be used as before, and any target variables are automatically monitored. Any variables specified using the inline `#data#` statement will be retrieved from the working environment as usual and will be common to all simulations - data which is intended to change between simulations must therefore be provided using the 'data function' argument instead. The 'run.jags.study' function can also be used to automate drop-k type validation studies, by specifying a data function that returns the same dataset for each simulation but with a different value (or values) made missing, and the full data as a target variable. An illustration of this function is provided in Section 3.

Large simulation studies are likely to be quite computationally intensive, but are an ideal candidate for parallelisation. For this reason, parallel computation is built directly into the 'run.jags.study' function using the **parallel** package. This can be used to parallelise the simulation locally, or using a high performance computing cluster set up using the **snow** package (Tierney, Rossini, Li, and Sevcikova 2013) and passed to the simulation study using the 'cl' argument. This allows the available computing power to be maximised without requiring any additional code to be written by the end user, including an initial check to ensure that the model compiles and runs locally (using a randomly chosen dataset) before beginning the parallelised study.

2.3. JAGS module

Besides the R code specified above, **runjags** also includes a modular extension to JAGS that provides users with access to an additional suite of functions for the Pareto family of distributions; extending the Pareto Type I distribution provided within JAGS itself to Pareto Types II, III, and IV distributions, as well as the generalised Pareto distribution, the Lomax distribution (a special case of the Pareto Type II distribution with $\mu = 0$), and a distribution advocated by DuMouchel (1994) for use with variance parameters (discussed in Section 3.2). The usage, PDF and lower bound for the support of each distribution is shown in Table 1 (all distributions have an upper bound of ∞ unless otherwise stated).

A shared library containing this module is installed within the **runjags** package, and will be loaded when either the 'module' argument contains 'runjags', or automatically when using any of the distributions or functions provided by the module. Alternatively, the module can

Name	Usage	Density	Lower
Pareto I [1]	`dpar1(alpha,sigma)` $\alpha > 0, \sigma > 0$	$\alpha\,\sigma^{\alpha}\,x^{-(\alpha+1)}$	σ
Pareto II	`dpar2(alpha,sigma,mu)` $\alpha > 0, \sigma > 0$	$\frac{\alpha}{\sigma}\left(\frac{\sigma + x - \mu}{\sigma}\right)^{-(\alpha+1)}$	μ
Pareto III	`dpar3(sigma,mu,gamma)` $\sigma > 0, \gamma > 0$	$\frac{\left(\frac{x-\mu}{\sigma}\right)^{\frac{1}{\gamma}-1}\left(\frac{x-\mu}{\sigma}^{\frac{1}{\gamma}} + 1\right)^{-2}}{\gamma\,\sigma}$	μ
Pareto IV	`dpar4(alpha,sigma,mu,gamma)` $\alpha > 0, \sigma > 0, \gamma > 0$	$\frac{\alpha\left(\frac{x-\mu}{\sigma}\right)^{\frac{1}{\gamma}-1}\left(\frac{x-\mu}{\sigma}^{\frac{1}{\gamma}} + 1\right)^{-(\alpha+1)}}{\gamma\,\sigma}$	μ
Lomax [2]	`dlomax(alpha,sigma)` $\alpha > 0, \sigma > 0$	$\frac{\alpha}{\sigma}\left(1 + \frac{x}{\sigma}\right)^{-(\alpha+1)}$	0
DuMouchel [3]	`dmouch(sigma)` $\sigma > 0$	$\frac{\sigma}{(x+\sigma)^2}$	0
Gen. Par.	`dgenpar(sigma,mu,xi)` $\sigma > 0$	$\frac{1}{\sigma}\left(1 + \xi\frac{x-\mu}{\sigma}\right)^{-\left(\frac{1}{\xi}+1\right)}$	μ [4]
		For $\xi = 0$: $\frac{1}{\sigma}\,e^{\frac{-(x-\mu)}{\sigma}}$	μ

[1] This is equivalent to the `dpar(alpha,c)` distribution and provided for naming consistency

[2] This is referred to as the '2nd kind Pareto' distribution by Van Hauwermeiren and Vose (2009); an alternative form for the PDF of this distribution is given by: $\frac{\alpha\,\sigma^{\alpha}}{(x+\sigma)^{\alpha+1}}$

[3] This distribution was suggested by DuMouchel (1994) as a suitable prior for τ in a Bayesian meta-analysis setting, and is equivalent to a Lomax distribution with $\alpha = 1$

[4] The Generalised Pareto distribution also has an upper bound of $x \leq \mu - \frac{\sigma}{\xi}$ for $\xi < 0$

Table 1: Distributions provided by the JAGS module included with **runjags**.

be loaded for use with the **rjags** package using the following command:

```
R> load.runjagsmodule()

Loading required package: rjags
Loading required package: coda
Loading required package: lattice
Linked to JAGS 3.4.0
Loaded modules: basemod,bugs
module runjags loaded
```

Use of the internal module is only available for the 'rjags' method (attempts to load the module will produce an error for other methods), however a standalone JAGS module containing the same functions is available from http://sourceforge.net/projects/runjags. The configuration of this module is based on the template given by Wabersich and Vandekerckhove (2013), and should therefore install on a variety of platforms using the standard './configure', 'make' 'make install' convention. Note that this module is named 'runjagsmodule' to avoid naming conflicts with the internal module, and the internal module will not be loaded automatically if the character vector provided to the 'module' argument contains 'runjagsmodule'.

3. Illustration of usage

Here we will consider a worked example of a simulation study analysis using **runjags**, to assess the sensitivity to various minimally informative priors of an over dispersed count model represented by multiple formulations of a negative binomial or gamma-Poisson compound distribution (for the sake of completeness, a proof of the equivalence of these distributions is included in Appendix A). Use of this distribution is widespread in many biological fields (Bolker, Brooks, Clark, Geange, Poulsen, Stevens, and White 2009), including parasitology (Wilson and Grenfell 1997; Wilson, Grenfell, and Shaw 1996; Shaw, Grenfell, and Dobson 1998), where Bayesian methods of analysis have been shown to provide more robust inference than traditional methods (Denwood, Stear, Matthews, Reid, Toft, and Innocent 2008; Denwood, Reid, Love, Nielsen, Matthews, McKendrick, and Innocent 2010). A pragmatic assessment of the sensitivity of this distribution to subtly different parameterisations is therefore merited.

3.1. Model formulation and assessment

The gamma distribution is parameterised in JAGS and BUGS by the `shape` (α) and `rate` (β) parameters, with the expectation given by $\frac{\alpha}{\beta}$ and variance given by $\frac{\alpha}{\beta^2}$. This distribution can be used to describe underlying variability in a Poisson observation, representing an unknown amount of over-dispersion between observations. In this situation the extra-Poisson coefficient of variation may be a more useful measure of the variability of the underlying gamma distribution Denwood (2010); this is represented by a function of the `shape` parameter: $\sqrt{\frac{1}{\alpha}}$

A candidate JAGS model (using inline data and monitor statements to be detected by **runjags**) is as follows:

```
R> jagsmodel <- "model{
+
+       for(i in 1:N){
+               Count[i] ~ dpois(lambda[i])
+               lambda[i] ~ dgamma(shape, rate)
+       }
+
+       shape ~ dgamma(0.001, 0.001)
+       mean ~ dgamma(0.001, 0.001)
+       rate <- shape / mean
+
+       #data# N
+       #inits# shape, mean, .RNG.seed, .RNG.name
+}"
```

This model allows each observed `Count` to follow a Poisson distribution with `lambda` drawn from a gamma distribution with `shape` parameter to be estimated, and `rate` parameter calculated from the `shape` parameter and the `mean` of the distribution, which is also to be estimated. The `Gamma(0.001,0.001)` distribution is a commonly used 'reference prior' for variance parameters such as the `shape` parameter of our gamma distribution; here we use the same minimally informative prior for both `shape` and `mean` parameters. The `#data#` statement is used to include `N` as data that does not change between simulations, and `#inits#` is used to include `shape` and `mean` as initial values for two chains. The `Count` variable is also observed, but will vary between simulations so is not retrieved from R memory using the `#data#` tag.

The performance of this model can be assessed using a simulation study, with data generated from a gamma-Poisson distribution with a mean of 2, shape parameter of 0.75, and a sample size of 20. These values are chosen to exaggerate any model performance issues by providing a comparatively small dataset with a large number of zero observations, and are similar to those typically found in veterinary parasitological datasets. This assessment can be automated using the 'run.jags.study' function, by creating a function to return some pre-generated simulated data, and then running the simulation study using a **snow** cluster of 20 processors (on the host machine) as follows.

Create simulated data and function to return one dataset per simulation:

```
R> N <- 20
R> S <- 1000
R> truemean <- 2
R> trueshape <- 0.75
R> truerate <- trueshape / truemean
R> set.seed(1)
R> alldata <- lapply(1:S, function(x){
+       return( rpois(N, rgamma(N, trueshape, rate = truerate) ) )
+ })
R> datafunction <- function(i)
+  return( list( Count = alldata[[i]] ) )
```

Set up initial values for 2 chains and a snow cluster (with 20 parallel threads on the local machine):

```
R> shape <- list(0.1, 10)
R> mean <- list(10, 0.1)
R> .RNG.seed <- function(chain)
+  return( switch(chain, "1"= 1, "2"= 2) )
R> .RNG.name <- function(chain)
+  return( switch(chain, "1" = "base::Super-Duper",
+  "2" = "base::Wichmann-Hill") )
R> library("parallel")
R> cl <- makeCluster(20)
```

Run the simulation study and show the results:

```
R> results <- run.jags.study(S, jagsmodel, datafunction,
+ targets = list( mean = truemean, shape = trueshape ),
+ runjags.options = list( n.chains = 2 ), cl = cl)

Starting a JAGS study at 01:17
Testing the model and data for simulation 990...
Compiling rjags model and adapting for 1000 iterations...
Finished running the simulation
The model runs OK
Calling autorun.jags for 1000 simulations...
Finished running the simulations
Finished summarising results
Finished JAGS study at 01:27 (total time taken: 6.9 minutes)

R> results

Average values obtained from a JAGS study with a total of 999 simulations
(excluding 1 crashed simulations):

      Target   Median     Mean Lower95%CI Upper95%CI Range95%CI Within95%CI
mean    2.00 2.037695 2.195683  0.9618164   3.732989   2.771172   0.9329329
shape   0.75 1.852515 6.025725  0.1719675  26.917416  26.745448   0.9579580
      AutoCorr(Lag10) Simulations
mean        0.1137136         999
shape       0.4883676         999

The 1 error returned has been stored in the '$errors' element of the list returned from ru

Average time taken:  5.2 seconds (range: 2.4 seconds - 36.9 seconds)
Average burnin required:  7272 (range: 5000 - 145000)
Average samples required:  10470 (range: 10000 - 57825)
```

The function returns an object of class 'runjags.study', with a default print method that summarises the results as appropriate; showing average values for the parameters that are common to multiple simulations, and individual results for parameters that are estimated from only a single simulation (for drop-1 cross validation studies). All individual simulations are run using the underlying autorun.jags function, which attempts to ensure that sufficient samples have been taken to ensure convergence and minimise Monte Carlo error.

For this simulation study, the `mean` parameter was estimated reasonably well, but there was a large positive bias for the median and mean estimates of the `shape` parameter which would result in under-estimation of the coefficient of variation. As would be expected, the 95% confidence intervals for both parameters identified the true value approximately 95% of the time. However, there was substantial autocorrelation for the shape parameter, which will have the effect of slowing convergence and increasing the required number of samples. One of the simulations also returned an error; further investigation revealed that the cause of the error ('Slicer stuck at value with infinite density') was sampling of extremely small values for the `shape` parameter for a dataset with a particularly large variance and small mean.

3.2. Sensitivity to prior distributions

There are various different minimally informative priors advocated for use with variance parameters in hierarchical models. The commonly used `Gamma(0.001,0.001)` distribution is characterised by a mean of 1 and a very large variance, and is almost improper (Gelman 2006). One alternative distribution that is guaranteed to be proper is a Uniform distribution with a lower bound of 0 and arbitrarily large upper bound, such as a `Uniform(0,1000)` distribution. For use with variance parameters in hierarchical models, DuMouchel (1994) proposed the use of a prior distribution with PDF given by:

$$\pi(\tau) = \frac{s_0}{(s_0 + \tau)^2}$$

Although this connection is not stated directly by the author, the PDF given above is equivalent to that of a Lomax distribution with $\tau = x$, $s_0 = \sigma$ and $\alpha = 1$, and therefore to a Pareto type II distribution with $\tau = x$, $s_0 = \sigma$, $\alpha = 1$ and $\mu = 0$ (Table 1). This DuMouchel distribution is guaranteed to be proper, and has a mode of zero with infinite mean and variance, which are conceptually desirable properties for a minimally informative prior. The choice of σ dictates the median of the distribution, with a value of 1 advocated because this also ensures invariance to the inverse transformation of τ, meaning that this prior can be equivalently applied to the variance or precision. If a different choice of σ is made, then the distribution of $\frac{x}{\sigma}$ is invariant to inverse transformation. Note that the Lomax (or Pareto Type II) distribution is not implemented directly in JAGS or BUGS, however the **runjags** module for JAGS implements this function as `dlomax(alpha,sigma)`, and also provides the function `dmouch(sigma)` that improves the computational efficiency of the underlying functions by removing the α parameter from the Lomax distribution. Alternatively, a dummy variable distributed according to a Pareto(1,1) distribution can be used with the true parameter calculated by subtracting one from this dummy variable using standard BUGS syntax - however this is only equivalent to the Lomax distribution for the parameterisation with $\alpha = 1, \sigma = 1$.

As a result of the desirable properties mentioned above, this distribution has been used as a minimally informative prior in situations outside the meta-analysis application for which

Mean Prior	Shape Prior	Mean	Range of CI	Within CI	Auto. Corr.	Simulations
Gamma	Gamma	2.20	2.77	0.93	0.11	999
Uniform	Gamma	2.67	4.24	0.95	0.24	996
Lomax	Gamma	2.09	2.50	0.93	0.08	999
Gamma	Uniform	2.13	2.31	0.86	0.27	1000
Uniform	Uniform	2.39	2.90	0.89	0.34	999
Lomax	Uniform	2.06	2.14	0.86	0.25	999
Gamma	Lomax	2.18	2.70	0.93	0.07	999
Uniform	Lomax	2.56	3.74	0.96	0.17	999
Lomax	Lomax	2.09	2.47	0.93	0.04	1000

Table 2: Average values for the inference on the mean parameter (true value 2) obtained from 1000 simulated datasets for nine gamma-Poisson MCMC models using three different minimally informative priors for the mean and shape parameters.

it was originally devised (for example Conti, Presanis, van Veen, Xiridou, Donoghoe, Rinder Stengaard, and De Angelis 2011; Yin, Conti, Desai, Stafford, Slater, Gill, and Simms 2013; Phillips, Tam, Conti, Rodrigues, Brown, Iturriza-Gomara, Gray, and Lopman 2010). Here, a pragmatic approach is taken to quantify the effect of some commonly used prior distributions on the inference made using the model specified in Section 3.1. Using every combination of `Gamma(0.001,0.001)`, `Uniform(0,1000)` and `Lomax(1,1)` distributions for the mean and shape parameters gives a total of nine candidate models. Each were run with the same 1000 simulated datasets, using very similar R code to that for the first model.

The results of these nine simulation studies are shown in Tables 2 and 3. Inference for the mean parameter was comparatively similar between the models, although use of a Uniform prior for the shape parameter resulted in poorly performing 95% confidence intervals and higher autocorrelation for the mean parameter. There was a marked difference between models in terms of the inference on the shape parameter, with a much larger bias and range of 95% CI for models using a gamma and particularly Uniform prior on the shape parameter compared to models using the Lomax prior for the shape parameter. The large bias of the Uniform prior is not surprising given that the mean of this distribution is 500, however a large positive bias was also observed with the Gamma prior with mean equal to 1. Autocorrelation was much higher for the shape parameter than the mean parameter for all models, with the lowest autocorrelation (and all datasets analysed successfully) using Lomax priors for both parameters.

3.3. Sensitivity of alternative model parameterisations

The model shown in Section 3.1 (hereafter denoted 'Model A') is not the only possible formulation of a gamma-Poisson distribution using the BUGS syntax. In Model A, the prior distributions were placed on the `mean` and `shape` parameter with the `rate` parameter calculated as a deterministic node from these, however it is equally possible to put the prior on the `rate` parameter and calculate the `mean` parameter deterministically (or even outside JAGS) as in Model B:

```
R> ModelB <- "model{
+         for(i in 1:N){
```

Mean Prior	Shape Prior	Mean	Range of CI	Within CI	Auto. Corr.	Simulations
Gamma	Gamma	6.03	26.75	0.96	0.49	999
Uniform	Gamma	5.86	25.36	0.95	0.49	996
Lomax	Gamma	5.81	25.66	0.96	0.48	999
Gamma	Uniform	115.42	255.49	0.92	0.59	1000
Uniform	Uniform	112.49	250.81	0.93	0.60	999
Lomax	Uniform	114.65	249.44	0.92	0.59	999
Gamma	Lomax	1.47	3.99	0.97	0.37	999
Uniform	Lomax	1.44	3.91	0.96	0.39	999
Lomax	Lomax	1.49	4.04	0.97	0.38	1000

Table 3: Average values for the inference on the shape parameter (true value 0.75) obtained from 1000 simulated datasets for nine gamma-Poisson MCMC models using three different minimally informative priors for the mean and shape parameters.

```
+                Count[i] ~ dpois(lambda[i])
+                lambda[i] ~ dgamma(shape, rate)
+        }
+
+        shape ~ dgamma(0.001, 0.001)
+        rate ~ dgamma(0.001, 0.001)
+        mean <- shape / rate
+
+        #data# N
+        #inits# shape, rate, .RNG.seed, .RNG.name
+}"
```

In this simple model, we could also formulate the model as a negative binomial distribution rather than a gamma mixture of Poisson distributions - the negative binomial distribution is parameterised by a probability `p` and a second parameter equivalent to the `shape` parameter (see Appendix A for more details). An alternative model formulation could therefore be represented as in Model C:

```
R> ModelC <- "model{
+
+        for(i in 1:N){
+                Count[i] ~ dnegbin(prob[i], shape)
+                prob[i] <- shape / (shape + mean)
+        }
+
+        shape ~ dgamma(0.001, 0.001)
+        mean ~ dgamma(0.001, 0.001)
+
+        #data# N
+        #inits# shape, mean, .RNG.seed, .RNG.name
+}"
```

Model	Prior	Mean	Range of CI	Within CI	Auto. Corr.	Simulations
Model A	Lomax	2.09	2.47	0.93	0.04	1000
Model A	Gamma	2.20	2.77	0.93	0.11	999
Model B	Lomax	2.06	2.33	0.92	0.01	1000
Model B	Gamma	3.87	2.78	0.94	0.03	999
Model C	Lomax	2.08	2.44	0.93	0.02	1000
Model C	Gamma	2.19	2.74	0.93	0.05	1000

Table 4: Average values for the inference on the mean parameter (true value 2) obtained from 1000 simulated datasets for three equivalent specifications of gamma-Poisson MCMC models, each using two sets of minimally informative priors for the mean and shape parameters.

Model	Prior	Mean	Range of CI	Within CI	Auto. Corr.	Simulations
Model A	Lomax	1.49	4.04	0.97	0.38	1000
Model A	Gamma	6.03	26.75	0.96	0.49	999
Model B	Lomax	1.25	2.67	0.98	0.57	1000
Model B	Gamma	2.51	8.60	0.96	0.65	999
Model C	Lomax	1.53	4.05	0.97	0.20	1000
Model C	Gamma	9.31	47.17	0.96	0.33	1000

Table 5: Average values for the inference on the shape parameter (true value 0.75) obtained from 1000 simulated datasets for three equivalent specifications of gamma-Poisson MCMC models, each using two sets of minimally informative priors for the mean and shape parameters.

These models are all intended to represent the same simple data structure, and share similar 'minimally informative' prior distributions on the two parameters of interest. A comparison of the posterior coverage and autocorrelation between these models can be made, using gamma priors and Lomax priors for both parameters to assess the relative sensitivity of the three models to the priors. This procedure is performed using the code given in Section 3.1 for these six candidate model formulations.

A comparison of the results from each model is shown in Tables 4 and 5. The two models A & B encountered an error from one dataset using gamma priors, but all datasets were analysed successfully using Lomax priors. Inference for the mean parameter was very similar between models A and C, but was somewhat different for Model B using the gamma prior. As before, inference for the shape parameter was heavily affected by the choice of prior distributions, with the models using Lomax priors out-performing the corresponding models using gamma priors for each model type. The choice of model formulation had a small effect on the inference made for the shape parameter using Lomax priors, particularly in terms of the autocorrelation and range of the 95'% CI produced, but a much larger effect on the inference made using gamma priors.

3.4. Discussion

The results presented here demonstrate the potential affect of multiple prior distributions, each of which could in some way be considered 'minimally informative', on the inference made from analysis of small over-dispersed count datasets. The Uniform prior would not

typically be used for a variance parameter, but is sometimes used for mean parameters, and is useful to demonstrate an extreme example of prior distribution influence. The results also show that a poor choice of prior distribution can also exaggerate differences in inference made by conceptually equivalent model formulations, and adversely affect the autocorrelation dependence within chains, reducing the effective sample size of a fixed length MCMC chain. DuMouchel's prior distribution (equivalent to a Lomax distribution) produced the least bias and autocorrelation for the simulated data analysed here; this may be partly a function of the specific characteristics of the data generated for this study, although a similar study using data generated with a mean of 1 and shape parameter of 10 produced similar results (data not shown). The desirable properties of this distribution, in terms of invariance to sampling the variance or precision, infinite mean/variance with a mode of 0, and being guaranteed to remain proper, suggest that the potential usefulness of this prior should be investigated for other applications (particularly for precision parameters). Although this distribution is not directly implemented in JAGS, the **runjags** package implements the full Pareto family of distributions using an extension module. Alternatively, it is relatively straightforward (although computationally slower) to obtain this distribution from a type I $Pareto(1, 1)$ distribution by subtracting one from a dummy variable.

4. Summary

Given the flexibility and ease of use of BUGS type software packages, the recent widespread adoption of these statistical tools to analyse data from a variety of disciplines is not surprising. The extendability of JAGS with user-specified modules is a major advantage compared to other implementations of BUGS, as it allows any arbitrary function or distribution to be implemented directly within JAGS, replacing the need for the 'ones' trick for using customised likelihood functions. A very useful tutorial on writing and installing a standalone JAGS module is provided by Wabersich and Vandekerckhove (2013), however it is substantially easier to configure and install a JAGS module as part of an R package using the **rjags** interface because many of the necessary environmental variables are set up by 'R CMD INSTALL' during installation of the package, leaving only a 'configure' file in the route directory and 'Makevars' file, specifying the required include files, libraries and make objects, to be specified by the user. The source code for the **runjags** package is freely available be used as a template for other users, and includes a brief overview of the process of writing a module within the README file.

There are huge advantages to using MCMC, but also some potential disadvantages associated with failure to identify poor convergence and high Monte Carlo error, as well as sensitivity of the inference made to using subtly different model specifications and prior distributions. The **runjags** package attempts to partially safeguard against some of these difficulties by calculating and automatically reporting convergence and sample length diagnostics every time a JAGS model is run, and providing a more user friendly way to access commonly used visual convergence diagnostics and summary statistics. The methods used to call the underlying executable are generalised and flexible, and can be used to parallellise model runs with multiple chains. A further attraction of this package is the facilitation of simulation studies, with analysis of an arbitrary number of replicate datasets fully automated and parallelised using a single function call.

There has been considerable debate in the literature concerning the relative merits of sub-

jective and objective Bayesian inference (see for example Berger and Berry (1988); Berger (2006); Lele and Dennis (2009)). Where prior information is available and can be justified, there can be no doubt that use of this information will have a beneficial effect on the posterior distributions of all parameters of interest - however, there are many applied fields where use of such prior information is often considered to be unjustified by the mainstream community. In these situations, it may be desirable to use a minimally informative prior, however there is always the potential for inference to be sensitive even to priors that are intended to be minimally informative, especially when dealing with small datasets. For over-dispersed count data with properties similar to that generated here, it appears that the prior distribution suggested by DuMouchel (1994) (a Lomax or Pareto Type-II distribution) is a suitable minimally informative prior, however the most appropriate prior for other applications is likely to be highly specific to that application. Embarking on a full simulation study to validate the model formulation used can be computationally demanding, but the availability of ever-increasing computing power brings these procedures well within reach. Where there is any doubt over the optimal model formulation for a particular application, a pragmatic and robust approach should therefore include some validation of the choice of model against simulated data.

References

Berger J (2006). "The Case for Objective Bayesian Analysis." *Bayesian Analysis*, **1**(3), 385–402.

Berger JO, Berry D (1988). "Statistical Analysis and the Illusion of Objectivity." *American Scientist*, **76**, 159–165.

Bolker BM, Brooks ME, Clark CJ, Geange SW, Poulsen JR, Stevens MHH, White JSS (2009). "Generalized Linear Mixed Models: a Practical Guide for Ecology and Evolution." *Trends in Ecology & Evolution*, **24**(3), 127–35. ISSN 0169-5347. URL http://dx.doi.org/10.1016/j.tree.2008.10.008.

Brooks SP, Gelman A (1998). "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics*, **7**(4), 434–455.

Brooks SP, Roberts GO (1998). "Assessing Convergence of Markov Chain Monte Carlo Algorithms." *Statistics and Computing*, **8**, 319–333.

Conti S, Presanis AM, van Veen MG, Xiridou M, Donoghoe MC, Rinder Stengaard A, De Angelis D (2011). "Modeling of the HIV Infection Epidemic in the Netherlands: a Multi-Parameter Evidence Synthesis Approach." *The Annals of Applied Statistics*, **5**(4), 2359–2384. ISSN 1932-6157. URL http://dx.doi.org/10.1214/11-AOAS488.

Denwood, MJ (In Review). "runjags: An R Package Providing Interface Utilities, Distributed Computing Methods and Additional Distributions For MCMC Models in JAGS." *Journal of Statistical Software*.

Denwood MJ (2010). *A Quantitative Approach to Improving the Analysis of Faecal Worm egg Count Data*. Doctoral thesis, University of Glasgow. URL http://www.gla.ac.uk/media/media%5C_149338%5C_en.pdf.

Denwood MJ, Reid SWJ, Love S, Nielsen MK, Matthews L, McKendrick IJ, Innocent GT (2010). "Comparison of Three Alternative Methods for Analysis of Equine Faecal egg Count Reduction Test Data." *Preventive Veterinary Medicine*, **93**(4), 316–23. ISSN 1873-1716. URL http://dx.doi.org/10.1016/j.prevetmed.2009.11.009.

Denwood MJ, Stear MJ, Matthews L, Reid SWJ, Toft N, Innocent GT (2008). "The Distribution of the Pathogenic Nematode *Nematodirus battus* in Lambs is Zero-Inflated." *Parasitology*, **135**(10), 1225–1235. ISSN 1469-8161 (Electronic). URL http://dx.doi.org/10.1017/S0031182008004708.

DuMouchel W (1994). "Hierarchical Bayes Linear Models for Meta-Analysis." *Technical Report 27*, National Institute of Statistical Sciences. URL http://www.niss.org/sites/default/files/pdfs/technicalreports/tr27.pdf.

Garthwaite PH, Kadane JB, O'Hagan A (2005). "Statistical Methods for Eliciting Probability Distributions." *Journal of the American Statistical Association*, **100**(470), 680–701. ISSN 0162-1459. URL http://dx.doi.org/10.1198/016214505000000105.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **1**(3), 515–533.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**(4), 457–472. URL http://www.jstor.org/stable/2246093.

Gilks WR, Richardson S, Spiegelhalter DJ (1998). *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Boca Raton, Fla. ISBN 0412055511. URL http://www.loc.gov/catdir/enhancements/fy0646/98033429-d.html.

Hastings WK (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika*, **57**(1), 97–109. URL http://dx.doi.org/10.1093/biomet/57.1.97.

Kass RE, Carlin BP, Gelman A, Neal RM (1998). "Markov Chain Monte Carlo in Practice: a Roundtable Discussion." *The American Statistician*, **52**(2), 93–100.

Kass RE, Wasserman L (1996). "The Selection of Prior Distributions by Formal Rules." *Journal of the American Statistical Association*, **91**(435), pp. 1343–1370. ISSN 01621459. URL http://www.jstor.org/stable/2291752.

Lele SR, Dennis B (2009). "Bayesian Methods for Hierarchical Models: are Ecologists Making a Faustian Bargain?" *Ecological Applications : a Publication of the Ecological Society of America*, **19**(3), 581–4. ISSN 1051-0761. URL http://www.ncbi.nlm.nih.gov/pubmed/19425420.

Lunn DJ, Thomas A, Best N, Spiegelhalter D (2000). "**WinBUGS** - a Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing*, **10**(4), 325–337. ISSN 0960-3174. URL http://dx.doi.org/10.1023/A:1008929526011.

Phillips G, Tam CC, Conti S, Rodrigues LC, Brown D, Iturriza-Gomara M, Gray J, Lopman B (2010). "Community Incidence of Norovirus-Associated Infectious Intestinal Disease in England: Improved Estimates Using Viral Load for Norovirus Diagnosis." *American Journal*

of Epidemiology, **171**(9), 1014–22. ISSN 1476-6256. URL http://dx.doi.org/10.1093/aje/kwq021.

Plummer M (2013a). *Just Another Gibbs Sampler (JAGS) Software, Version-3.4.0*. URL http://mcmc-jags.sourceforge.net.

Plummer M (2013b). ***rjags**: Bayesian Graphical Models Using Mcmc*. URL http://cran.r-project.org/package=rjags.

Plummer M, Best N, Cowles K, Vines K (2006). "**CODA**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL http://cran.r-project.org/doc/Rnews/.

Polson NG, Scott JG (2011). "On the Half-Cauchy Prior for a Global Scale Parameter." *Cornell University Library: arXiv.org*. URL http://arxiv.org/abs/1104.4937.

Shaw DJ, Grenfell BT, Dobson AP (1998). "Patterns of Macroparasite Aggregation in Wildlife Host Populations." *Parasitology*, **117**, 597–610. ISSN 0031-1820.

Tierney L, Rossini AJ, Li N, Sevcikova H (2013). ***snow**: Simple Network of Workstations*. URL http://cran.r-project.org/package=snow.

Toft N, Innocent GT, Gettinby G, Reid SWJ (2007). "Assessing the Convergence of Markov Chain Monte Carlo Methods: an Example from Evaluation of Diagnostic Tests in Absence of a Gold Standard." *Preventive Veterinary Medicine*, **79**(2-4), 244–256. ISSN 0167-5877. URL http://dx.doi.org/10.1016/j.prevetmed.2007.01.003.

Tuyl F, Gerlach R, Mengersen K (2008). "A Comparison of Bayes–Laplace, Jeffreys, and Other Priors." *The American Statistician*, **62**(1), 40–44. ISSN 0003-1305. URL http://dx.doi.org/10.1198/000313008X267839.

Van Hauwermeiren M, Vose D (2009). *A Compendium of Distributions*. Vose Software, Ghent, Belgium. URL http://www.vosesoftware.com/content/ebook.pdf.

Wabersich D, Vandekerckhove J (2013). "Extending JAGS: a Tutorial on Adding Custom Distributions to JAGS (With a Diffusion Model Example)." *Behavior Research Methods*. URL http://www.cidlab.com/prints/wabersich2013extending.pdf.

Wilson K, Grenfell BT (1997). "Generalized Linear Modelling for Parasitologists." *Parasitology Today*, **13**(1), 33–38. ISSN 0169-4758. URL http://dx.doi.org/10.1016/S0169-4758(96)40009-6.

Wilson K, Grenfell BT, Shaw DJ (1996). "Analysis of Aggregated Parasite Distributions: a Comparison of Methods." *Functional Ecology*, **10**, 592–601.

Yin Z, Conti S, Desai S, Stafford M, Slater W, Gill ON, Simms I (2013). "The Geographic Relationship Between Sexual Health Deprivation and the Index of Multiple Deprivation 2010: a Comparison of two Indices." *Sexual Health*, **10**(2), 102–11. ISSN 1448-5028. URL http://dx.doi.org/10.1071/SH12057.

A. Formulation of the negative binomial as a gamma-Poisson

The compound probability mass function of a Poisson distribution (with mean λ) integrated over a gamma distribution (with shape and scale parameters α and β respectively) is given in Equation 1.

$$f(x;\alpha,\beta) = \int_0^\infty \frac{\lambda^x}{x!}e^{-\lambda} \,.\, \beta^\alpha \frac{1}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}\,\mathrm{d}\lambda \tag{1}$$

Substituting $\alpha = r$ and $\beta = \frac{1-p}{p}$ into Equation 1 gives Equation 2, which can be re-written and simplified to Equation 4.

$$f(x;r,p) = \int_0^\infty \frac{\lambda^x}{x!}e^{-\lambda} \,.\, \left(\frac{1-p}{p}\right)^r \frac{1}{\Gamma(r)}\lambda^{r-1}e^{-\left(\frac{1-p}{p}\right)\lambda}\,\mathrm{d}\lambda \tag{2}$$

$$= \frac{(1-p)^r}{x!\,p^r\,\Gamma(r)} \int_0^\infty \lambda^{x+r-1}e^{-\lambda}e^{-\frac{(1-p)\lambda}{p}}\,\mathrm{d}\lambda \tag{3}$$

$$= \frac{(1-p)^r}{x!\,p^r\,\Gamma(r)} \int_0^\infty \lambda^{x+r-1}e^{-\frac{\lambda}{p}}\,\mathrm{d}\lambda \tag{4}$$

Substituting the gamma function $\frac{\Gamma(b+1)}{a^{b+1}} = \int_0^\infty t^b e^{-at}\mathrm{d}t$ for $a = \frac{1}{p}$, $b = x + r - 1$ and $t = \lambda$ into Equation 4 gives Equation 5.

$$f(x;r,p) = \frac{(1-p)^r}{x!\,p^r\,\Gamma(r)} \frac{\Gamma(x+r-1+1)}{\left(\frac{1}{p}\right)^{x+r-1+1}} \tag{5}$$

$$= \frac{(1-p)^r}{x!\,p^r\,\Gamma(r)}\,\Gamma(x+r)\,p^{x+r} \tag{6}$$

$$= \frac{\Gamma(x+r)}{x!\,\Gamma(r)}\,p^x(1-p)^r \tag{7}$$

Equation 7 is the probability mass function of the negative binomial distribution defining the number of successes x before r failures with a probability of success p, which is therefore exactly equivalent to a gamma-Poisson compound distribution with mean $\frac{\alpha}{\beta} = \frac{pr}{1-p}$ and shape parameter $\alpha = r$.

Affiliation:

Matthew J Denwood
School of Veterinary Medicine
College of Medical, Veterinary and Life Sciences
University of Glasgow
UK
E-mail: matthew.denwood@glasgow.ac.uk
URL: http://www.gla.ac.uk/boydorr/people/byname/matthewdenwood/