Desirable BUGS in models of infectious diseases

Megan Auzenbergs\(^a,b\), Carla Correia-Gomes\(^c\), Theo Economou\(^d\), Rachel Lowe\(^b,e,f\), Kathleen M O’Reilly\(^a,b,⁎\)

\(^a\) Faculty of Infectious and Tropical Disease, London School of Hygiene and Tropical Medicine, London, UK
\(^b\) Centre for the Mathematical Modelling of Infectious Diseases, London School of Hygiene and Tropical Medicine, London, UK
\(^c\) Epidemiology Research Unit, SRUC, Edinburgh, UK
\(^d\) College of Life and Environmental Sciences, University of Exeter, Exeter, UK
\(^e\) Department of Infectious Disease Epidemiology, London School of Hygiene and Tropical Medicine, London, UK
\(^f\) Barcelona Institute for Global Health (ISGlobal), Barcelona, Spain

**ARTICLE INFO**

**Abstract**

Bayesian inference using Gibbs sampling (BUGS) is a set of statistical software that uses Markov chain Monte Carlo (MCMC) methods to estimate almost any specified model. Originally developed in the late 1980s, the software is an excellent introduction to applied Bayesian statistics without the need to write a MCMC sampler. The software is typically used for regression-based analyses, but any model that can be specified using graphical nodes are possible. Advanced topics such as missing data, spatial analysis, model comparison and dynamic infectious disease models can be tackled. Three examples are provided: a linear regression model to illustrate parameter estimation, the steps to ensure that the estimates have converged and a comparison of run-times across different computing platforms. The second example describes a model that estimates the probability of being vaccinated from cross-sectional and surveillance data, and illustrates the specification of different models, model comparison and data augmentation. The third example illustrates estimation of parameters within a dynamic Susceptible-Infected-Recovered model. These examples show that BUGS can be used to estimate parameters from models relevant for infectious diseases, and provide an overview of the relative merits of the approach taken.

1. Introduction

BUGS is a software for Bayesian inference using Gibbs sampling (Gilks et al., 1994). The software is now in its third decade, and has undergone several developments in its use and application. Although the software is sufficiently generic that it can be used within many data-driven fields, perhaps due to the affiliations of its developers BUGS is often used in medical sciences, but has also been widely used in social sciences, ecology and environmental sciences.

For a full description of the developments of BUGS see the article by (Lunn et al. (2009)) and the associated commentaries at the end of the article. The rationale behind developing BUGS was a need to make Bayesian analysis more accessible. Whilst the 1970–2000s saw many developments in Bayesian analysis, markov chain monte carlo (MCMC) analysis was largely restricted to models in closed form where a conjugate prior was required for specification of the model (where a conjugate prior is part of the same family of probability distributions as the posterior). A simple example of using conjugacy to estimate parameters from a model is the estimation of probability of occurrence from data. The likelihood is assumed to be binomially distributed where the data consists of k successes from n trials, \(Pr(x = k|p,n) = \binom{n}{k} p^k (1-p)^{n-k}\). To estimate the posterior distribution of the probability of success (p) we first specify the posterior from Bayes rule; \(Pr(p|n,k) \propto Pr(n,k|p)Pr(p)\) where \(Pr(p)\) is assumed to be a beta prior with parameters \(\alpha\) and \(\beta\). The probability density function of a beta distribution is \(Pr(p) = p^\alpha (1 - p)^{\beta - 1}\). Conjugacy occurs in this circumstance because the prior and posterior have the same distributional form, and the posterior can be sampled using \(p \sim B(\alpha + k, \beta + n - k)\) as \(Pr(p|n,k) \propto p^\alpha k (1 - p)^{\beta + n-k}\) (Rice, 2007). However, a closed form posterior distribution is unusual for most problems and additional (often impractical) mathematical manipulation is required to identify the posterior distribution, which prevents widespread use. The solution developed by Lunn et al. (2009) makes use of graphical modelling theory (Belliot, 2016), and the development of the BUGS language to specify models. The network of
nodes define the model where each node is either data or a parameter, and the edges between each node define the dependencies between the nodes. The dependencies illustrate the conditional probabilities assumed between nodes (which are usually directed), and is a core element of Bayesian inference. Additionally, the specification of directed acyclic graphs (DAGs) and automated translation to code means that scientists without a statistics or programming background are able to develop their own models.

As opposed to other languages that require considerable translation of equations into code (Handel, 2017), the language used to specify models in BUGS has a much lower learning curve for scientists to translate theory into practice. The original WinBUGS software has been available since the 1990s (Gilks et al., 1994). Programming developments, applications and interest within the scientific community has grown steadily since. The estimation procedures within the software expanded from using a Gibbs sampler to a self-tuning Metropolis updater, to increase the flexibility of the full conditional probability that can be specified and increase the efficiency of the estimation procedure. Additional modules were developed for specific applications; PkBUGS for application to pharmacokinetic models (Lunn et al., 2002) and the associated complex functions, and GeoBUGS for spatial modelling and the use of structured random errors (Thomas et al., 2004). Over 30 years of development has led to multiple software platforms performing very similar tasks. In the early 2000s clones and suitable alternatives of the original software were developed; first OpenBUGS (Thomas et al., 2006) and then JAGS (just another gibbs sampler (Plummer, 2003)), both of which facilitated use of the software by linux and MacOS users. To make use of multi-core processors (common to most computers) and reduce the run-time of MCMC estimation, multiBUGS was released in 2017 (Goudie et al., 2017). Small but important differences between them (Table 1) mean that all versions are likely to be in use for the foreseeable future. Integration with other software such as R is facilitated by calling the software within bespoke libraries (e.g. BRugs (Thomas et al., 2006), R2WinBUGS (Sturtz et al., 2005), runJAGS (Denwood, 2016)). Development of additional custom distributions within JAGS is possible and requires a working knowledge of C++ (Wabersich and Vandekerckhove, 2014).

There are now a vast number of worked BUGS examples and applications, which are available via affiliated websites, tutorials, peer-reviewed papers and books. Most applications are centred on the analysis of data where variation in the response requires explanation. Examples include the classic linear and generalised linear model analysis of data where variation in the response requires explanation. Selection of informative as they can encompass a wide range of values and consequently the posterior will largely be informed by the data. Selection of appropriate priors can be a challenging process and it is important to examine priors and understand the influence of priors on the posterior distribution (Seaman et al., 2012). Additionally, specification and estimation of the standard deviation is often more intuitive than use of precision for a parameter, and additional code may be used to specify the standard deviation instead of precision. This model can be written either as a DAG (Fig. 1) or directly within the BUGS language;

### 2. Case studies

#### 2.1. A linear model, associated output and comparison of run time between software

##### 2.1.1. Model specification

A simple linear regression model assumes that the response variable \( Y = (Y_1, \ldots, Y_n) \) is normally distributed with mean \( \mu \) and precision \( \tau \) (ie. \( Y_i \sim N(\mu, \tau) \)). We assume that an explanatory variable \( X = (X_1, \ldots, X_n) \) explains some of the variation in \( Y \). We assume that the model takes the form \( \mu = \alpha + \beta X \), which introduces two additional parameters that require estimation. Within a Bayesian setting priors are assigned to these parameters; both are assumed to be normally distributed with mean 0 and precision 0.5, which can be written as \( \alpha \sim N(0,0.5) \) and \( \beta \sim N(0,0.5) \). These priors are regarded as minimally informative as they can encompass a wide range of values and consequently the posterior will largely be informed by the data. Selection of appropriate priors can be a challenging process and it is important to examine priors and understand the influence of priors on the posterior distribution (Seaman et al., 2012). Additionally, specification and estimation of the standard deviation is often more intuitive than use of precision for a parameter, and additional code may be used to specify the standard deviation instead of precision. This model can be written either as a DAG (Fig. 1) or directly within the BUGS language;

#### Table 1

<table>
<thead>
<tr>
<th>Software</th>
<th>First available</th>
<th>Operating system</th>
<th>Extensions</th>
<th>JAGS First available</th>
<th>multiBUGS First available</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinBUGS</td>
<td>Mid 1990s</td>
<td>MS Windows</td>
<td>PkBUGS, glm, GeoBUGS</td>
<td>2008</td>
<td>2017</td>
</tr>
<tr>
<td>OpenBUGS</td>
<td>2006</td>
<td>MS Windows/Linux/Mac&lt;sup&gt;a&lt;/sup&gt;</td>
<td>GeoBUGS, glm, MultiBUGS</td>
<td>MS Windows, Mac, Linux</td>
<td>Windows (Linux under development)</td>
</tr>
</tbody>
</table>

<sup>a</sup> But note that OpenBUGS hasn’t been fully tested within the Mac OS. <sup>b</sup> GeoBUGS has not yet been fully tested within JAGS.
The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

```r
model{
  for(i in 1:N){
    y[i] ~ dnorm(mu[i],tau)
    mu[i] <- alpha + beta*x[i]
  }
  alpha ~ dnorm(0,0.5)
  beta ~ dnorm(0,0.5)
  log.sigma ~ dunif(0,100)
  sigma <- exp(log.sigma)
  sigma.sq <- pow(sigma,2)
  tau <- 1/sigma.sq
}
```

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.

The code is deliberately similar to the mathematical equations, creating a natural bridge from equations to code and vice versa. Note that BUGS specifies a normal distribution using the mean ($\mu$) and precision ($\tau$). For a generalised linear model structure it is also possible to specify the model using the standard lme-4 style syntax within R.
through loading the runjags package (Denwood, 2016); model <- template.jags(y ~ x, data, n.chains = 3, family = ‘gaussian’).

Continuing with the linear model example above, this model is used to generate simulated data with sample sizes ranging from 100 to 10,000 to examine the differences between software and platforms. In MS Windows the models were run on a 4-core machine consisting of 3.40 GHz and 8 processors with 16GB of RAM. In the Mac operating system the models were run on a 3.2 GHz Intel core i5 machine with 4 processors consisting of 16GB of RAM. Estimation of the posterior distribution was implemented by specifying the number of MCMC iterations, the number of initial iterations that will be discarded (ie. burn-in), the extent of thinning (ie. extracting every jth iteration of the MCMC), and the number of MCMC chains. The general principal of specifying these settings is to obtain a stationary target posterior distribution. There is no certain way to assess when the stationary distribution has been reached, but rather there are techniques to establish when it has not been reached (Toft et al., 2007). It is ideal to obtain the equivalent of 1000 independent samples of the posterior distribution, and given that autocorrelation of MCMC chains is common, generating at least 10,000 samples per chain should be considered a minimum value. Some software provides estimates of the effective sample size which provides an estimate of the equivalent number of independent samples (for example the coda package in R (Plummer et al., 2006)), and this value should exceed 1000 for all parameters. Chain convergence can be assessed visually by plotting the sampled value against its number in the chain. Running several chains with different starting values and comparing the sampled values on the same figure will illustrate whether enough burn-in has been specified (more noise in early iterations may be identified) and that the chains have converged to a common mean value, if convergence has been achieved. To assess convergence, the Multivariate Potential Scale Reduction Factor (also known as the Gelman-Rubin statistic (Brooks and Gelman, 1998)) can be applied. The Gelman-Rubin statistic compares the variance between the chains to the variance within the chains of each parameter, and if these are similar (indicating convergence) then its value should be less than 1.05. Autocorrelation plots can be used to assess the extent of autocorrection in MCMC chains and further inform the extent of thinning that needs to be specified.

2.1.2. Model results and interpretation

For this example each model was specified in an identical form and 15,000 MCMC iterations were run within each of three chains. The first 5000 iterations were regarded as ‘burn-in’ and discarded. The Gelman-Rubin statistic was applied to an initial round of samples where values were < 1.01 which suggests that the between-chain variance is low and consistent with convergence of the chains (Fig. 2). The autocorrelation plot illustrated a lag to approximately 5, so the model was re-run (10,000 iterations with 5000 burn-in) and the output was thinned to every 10th iteration. The time-series and density plots of the subsequent output (Fig. 2) illustrate consistent values across the chains. The effective sample size of the 10,000 iterations was at least 1435.

Table 2 illustrates that the run-time for each version of BUGS is linear with the size of the data for WinBUGS, JAGS and OpenBUGS. Whilst the run time also increases with models that have more parameters, the size of the data is usually the limiting factor for medical problems (especially when considering that the number of parameters should be much less than the number of observations within data). The relatively slow run time associated with the MCMC estimation has perhaps limited more widespread use of BUGS (Cowles, 2004), but these issues are not unique to BUGS but are common to most MCMC estimation approaches. Recent developments of the R library Nimble enables conversion of BUGS code to C++ code, which in the example above, has improved the run time by a factor of approximately 25. Inclusion of Nimble into the model construction adds additional programming complexity, but a suitable model can be developed within BUGS on a subset of a large dataset, and once developed, then specified within Nimble. Using multiBUGS to run the models resulted in a faster run time than JAGS, openBUGS and winBUGS and did not require any additional coding.

2.2. Spatio-temporal analysis of data: estimating vaccine effectiveness

Analysing polio vaccination data from Ethiopia shows how BUGS is particularly useful for combining multiple data sources into a statistical model. The aim of the analysis was to estimate the probability that a child aged 12 months would be vaccinated with the oral polio vaccine (OPV), and examine whether there was evidence for spatial or temporal variation in this estimate of vaccination coverage. Three different data sources were used: (1) Demographic and Health Survey (DHS) 2011; (2) DHS 2016; and (3) non-polio Acute Flaccid Paralysis (AFP) surveillance data for 2005–2016 from the WHO Polio Information System (Tangermann et al., 2017). Within the DHS data, the number of OPV doses were reported for each child aged < 5 years of age included in each cross-sectional survey. For the non-polio AFP data, cases of non-polio AFP were assumed to be an opportunistic sample of children aged < 5 years of age within Ethiopia where OPV vaccination histories were recorded as part of the case investigation. The non-polio AFP data have been previously used to estimate country- and province-level probabilities of being immunised with 3+ doses, where higher values were previously associated with a lower probability of reporting poliomyelitis outbreaks (O’Reilly et al., 2017; Tegegne et al., 2018), but its predictive ability may be affected by the uncertainty in the estimates. With the addition of DHS data estimates of vaccine effectiveness are likely to be more representative and reliable.

2.2.1. Model formulation

The reported number of OPV doses were converted to a response variable of whether 3 + OPV doses had been received (yi,d) where i refers to a child’s index and d refers to the dataset origin, which is omitted from further equations for clarity), and explanatory variables included in the model were year of vaccination according to the Gregorian calendar (t), which was inferred from the child date of birth and that routine OPV doses in Ethiopia are administered at 6, 10 and 14 weeks (WHO, 2019)), district of residency (z), number of eligible supplementary immunisation activities (SIAs, si) extrapolated from an OPV SIA calendar and exposure to routine immunisation (r) using Diphtheria-Tetanus-Pertussis (DTP) vaccination information (the DTP vaccine is administered concurrently with OPV drops in the routine immunisation series). Dose history of DTP vaccination was included within the DHS surveys and was used to augment DTP vaccination data for AFP cases, as DTP vaccination is not included in the AFP surveillance data in Ethiopia.

The model was used to test the hypothesis that estimates of vaccination coverage vary across districts and in time and that SIAs increase the probability of a child being ‘fully vaccinated’ (i.e. receiving 3 + OPV doses). Additionally, each data source is assumed to have an associated reporting factor to account for small changes in how the survey question is asked, differences in the sampled populations and
potential reporting bias (Cutts et al., 2013), modelled using an adjustment factor \( \beta_p \). We assume a binomial model for the response with associated regression coefficients as described above and mechanistic variables that describe individual vaccination histories from the data. The response variable \( Y_i \) takes the value of 0 or 1 according to whether 3 + OPV were reported by data source (\( Y_3 \)) and routine immunisation. The parameters \( \beta_3 \) and \( \beta_p \) correspond to the parameters associated with SIAs and DTP OPV doses, \( \hat{\beta}_3 \) and \( \hat{\beta}_p \) are the corresponding variables for the covariates. The effectiveness (ie. the probability of receiving 3 + OPV doses associated with each incremental increase in SIA or DTP) is \( 1 - (1/e^{\hat{\beta}_p}) \) and \( 1 - (1/e^{\hat{\beta}_3}) \).

The model is as follows:

\[
Y_i \sim \text{binomial}(\mu_i, 1)
\]

\[
\logit(\mu_i) = \hat{\beta}_3 z_i + \hat{\beta}_p t_i + \hat{\beta}_d d_i + \hat{\beta}_s \text{si}a_i + \hat{\beta}_d \text{dtp}_i
\]

Developing the model in BUGS allowed for changes in the model structure such as inclusion of interaction terms and adaptation of the model beyond a standard generalised linear model framework to be made with relative ease. For example, data on OPV doses provided via routine immunisation was not available within the AFP dataset so it was augmented (Kery and Schaub, 2011) from the spatial-temporal patterns in the DHS data assuming,

\[
d\text{dtp}_i \sim \text{binomial}(\mu_{ij}, 1)
\]

\[
\logit(\mu_{ij}) = a_i t_i - y_i \text{z}_i \cdot \log(\text{age}_i)
\]

To complete the model, we specify minimally informative priors where the regression coefficients were assigned \( \beta_3, \beta_p, \beta_d, \beta_s, \beta_d \sim N(0, \tau) \) and \( \tau \sim \text{U}(0,1.1) \) so that the posterior was largely influenced by the data. In this circumstance selection of alternative priors resulted in consistent posterior distributions but sometimes the posterior is unidentifiable and different variances were selected [Lawson, 2009]. The model was implemented in JAGS and the outputs examined within R. We generated three MCMC chains of length 10,000 iterations with a burn-in of 5000 and thinning to every 10th to obtain 1500 samples from the joint posterior distribution. To transform the model parameters into more interpretable outputs, the parameters were used to estimate the district-level probabilities of receiving 3 + doses of OPV through routine immunisation and the probability of a child 12 months of age receiving 3 + OPV doses through both routine immunisation and scheduled SIAs for 2011, along with estimates of vaccine effectiveness.

Different models were run in order to assess district estimates of vaccination in the presence of different covariates. The deviance information criteria (DIC) was used to compare the fit of each model and the model with the smallest DIC was assumed to provide the best fit to the data. At least two runs of the model were generated to compare the DICs to ensure consistency of the outputs.

2.2.2. Model results and interpretation

A model was developed to account for district of residency and year of vaccination as explanatory variables, and was compared to including the impact of the number of SIAs (by adding \( \beta_d \)) and whether this impact varied by district, year, or both. The DIC values illustrate increased evidence for inclusion of the SIA exposure histories into the models (DIC 22,679.3 compared to 22,842.8, respectively). Further model developments included augmented data so it was not possible to directly compare DIC values. Outputs of the probability of being vaccinated via routine immunisation and overall vaccination probabilities show that OPV vaccination coverage in Ethiopia varies spatially (Fig. 3), and vaccination has steadily improved since 2012 (Table 3). The effectiveness of the first SIA was estimated to be 0.44 (95% CI 0.34-0.53), and subsequent SIAs were estimated to further improve the chances of being fully vaccinated but with diminishing returns (DIC of constant effectiveness model = 36,348.8 vs. DIC of per-dose model = 36,285.9, difference = 62.9). The model also suggests that the source of vaccination data impacts estimates of vaccination coverage, with DHS data typically reporting a lower odds of a child 12 months of age being vaccinated with 3+ doses of OPV than the AFP data (odds ratio associated with DHS data 0.35 (95% CI 0.32–0.38)).

2.2.3. Comments as a first time user

Building the model in JAGS was relatively straightforward and existing example code (from similar applications) was easily replicable, which made the process behind building the model easier. Because adjusting priors and parameters within the JAGS model could be done with ease, the model could be built in a stepwise manner for each dataset by adding one covariate and corresponding prior at a time until the final model with all the data was constructed. Once all data was added to the final model, it was straightforward to parameterise the model and easy to apply the aforementioned minimally informative priors. In an effort to fit the best model to the data, a conditional autoregressive model (CAR model (McCarthy, 2009)) was trialed using spatial adjacency data and implemented in OpenBUGS (as the GeoBUGS module has not yet been fully tested in JAGS). This model takes into account spatial autocorrelation between neighbouring areal units and uses a spatial covariance matrix to assess spatial correlation that cannot be explained by the other model covariates alone, assuming that \( v[1:N] \sim \text{car.normal(adj[,],weights[,]},\text{num[,]},\tau_r) \) where \( \text{adj} \) is a spatial matrix describing the neighbourhood structure, \( \text{weights} \) is the corresponding weights for the neighbourhood structure, \( \text{num} \) is the sum of all neighbours and \( \tau_r \) is the standard deviation. Here, the adjacency weights were taken to be simple binary values; 1 if district \( d_i \) has a common boundary with \( d_j \) and 0 otherwise.

In comparison to the model implemented in JAGS, making changes to the openBUGS model was more difficult. Priors needed adjusting each time a new covariate or parameter was added into the model (to prevent the model from crashing) and the run time was much longer than when run in JAGS. OpenBUGS trap windows pop up each time an unsatisfactory model is run and deciphering the convoluted error messages can be difficult and time consuming to amend. JAGS errors appear directly in the R Console and contain more constructive feedback, such as indicating exactly which line of code contains the error. We found the model without the CAR structure had a much better fit to the data (difference in DIC > 100) so the CAR model was discarded.

2.3. Infectious disease dynamics: estimating transmission from outbreak data

Mechanistic (as opposed to statistical) transmission models are extremely useful in understanding disease dynamics. In particular, the class of Susceptible-Infectious-Resistant (SIR) models are widely used to estimate transmission and test the efficacy of control measures. SIR models describe the spread of infectious disease through a population in time, and the extent of spread depends on natural history parameters such as the transmission rate and the duration of infectiousness (Keeling and Rohani, 2008; Renshaw, 1991). Such parameters are traditionally assumed known, however with appropriate (i.e. detailed) data, these can be estimated exploiting the flexibility of Bayesian modelling to allow for flaws in the data, and to fully quantify the associated uncertainty.

Here, we consider data on occurrence of Salmonella typhimurium in pigs, using simulated data based on (Correia-Gomes et al. (2014)). The data consists of bi-weekly counts (over 18 weeks so that \( t = 1,\ldots,9 \)) of animals that are either classified as susceptible (S) or infectious (I) or resistant/carerrier (R), for 8 pig cohorts. These classifications of infection state were based on imperfect tests, a point we return to later.
2.3.1. Model formulation

Most conventional SIR models assume that the rate at which an animal has infectious contacts is constant in time and proportional to the density of infectious animals. The constant of proportionality $\beta$ is called the transmission rate parameter. Assuming contacts between animals are random, the number of infections in a bi-weekly time step is Poisson distributed with mean $\lambda_t = \beta \left( \frac{I_t}{N_t} \right)$, where $N_t$ is the total number of animals. The probability of no infectious contacts per animal is then $\text{exp}(-\beta \left( \frac{I_t}{N_t} \right))$ so that the probability of infection is $p_t = 1 - \text{exp}(-\beta \left( \frac{I_t}{N_t} \right))$. The number of new cases assumed to be $C_t \sim \text{Binomial}(S_t, p_t)$. To allow for cohort variability ($j = 1, \ldots, 8$) as well as temporal variation due to external factors, we include a zero mean random cohort-time effect, so that $p_{jt} = 1 - \text{exp}(-\beta \left( \frac{I_{jt}}{N_{jt}} \right)) \text{exp}(r_{jt})$. As such, we can formulate the model for new infections at the end of time period $t$ as $C_{jt} \sim \text{Binomial}(S_{jt}, p_{jt})$.

$$p_{jt} = 1 - \text{exp}(-\beta \left( \frac{I_{jt-1}}{N_{jt-1}} \right) \text{exp}(r_{jt}))$$

$$c \log\log(p_{jt}) = \log(\beta) + \log(I_{jt-1}) - \log(N_{jt-1}) + r_{jt}$$

$$r_{jt} \sim N(r_{j,t-1}, \sigma^2) \text{ for } t > 1, j = 1, \ldots, 8$$

$$r_{j1} \sim N(0, 100)$$

So that for each cohort, $j$, the effects $r_{jt}$ capture any unobserved but structured (modelled by a random walk) effects in time.

Recall however, that the detection of the infection was based on imperfect tests (serological and bacteriological), whose parallel specificity (true negatives) can nonetheless be assumed 100%. However, the
sensitivity (true positives) of both tests is not 100% and as such the observed number of infected animals $I_{obsjt}$, are a lower bound of the true (unobserved) number $I_{jt}$. We can then write $I_{jt} = I_{obsjt} + I_{objt}$ where $I_{objt}$ is the number of false positives. This can be modelled as: 

$$I_{objt} \sim \text{Binomial}(N_{jt}, p_{ND})$$

Where the probability of not detecting an infected case $p_{ND} = (1 - \text{SenC})(1 - \text{SenE})$, with \text{SenC} and \text{SenE} representing the sensitivity probability of each test (and perfect specificity (Eriksson and Aspan, 2007; Harris, 2003)). Treating $I_{objt}$ as unobserved allows formal quantification of the uncertainty due to the test sensitivity. Historical information on the sensitivity of both tests ((Eriksson and Aspan, 2007; Harris, 2003)) was used to construct informative beta prior distributions for \text{SenC} and \text{SenE}, namely $\text{SenC} \sim \text{Beta}(48.5, 50.5)$ and $\text{SenE} \sim \text{Beta}(58.5, 27.5)$.

To model the transition from infectious to resistant we can use similar arguments as before (see (Correia-Gomes et al. (2014)) for details) and write 

$$R_{newjt} \sim \text{Binomial}(I_{jt}, p_{Rj})$$

$$\text{cloglog}(p_{Rj}) = \log(\alpha) + sj$$

$$sj \sim N(0, \sigma_s^2)$$

Where $R_{newjt}$ is the number of (new) animals that become resistant at the end of $t$, and $p_{Rj}$ is the probability that an animal becomes resistant. Parameter $\alpha$ is the associated recovery rate, while $sj$ is a cohort random effect allowing for cohort heterogeneity.

Finally, considering the transition from resistant to infectious we use the following model

$$I_{newjt} \sim \text{Poisson}(\mu_{jt})$$

$$\log(\mu_{jt}) = \log(\nu) + \log(R_{jt-1}) + q_j$$

$$q_j \sim N(0, \sigma_q^2)$$

again using similar arguments to before, but replacing the binomial with a Poisson on the basis that this transition is a very rare event and that a Poisson distribution is a good approximation to the binomial distribution for small values of $p$ (Rice, 2007). $I_{newjt}$ is the number of new infectious animals (from this transition) in cohort $j$ at end of time $t$, while $\nu$ is the transmission rate parameter from resistant to infectious. $R_{jt-1}$ is the number of resistant animals in the previous time step, so its logarithm is used as an offset. Lastly, $q_j$ is another cohort random effect.

2.3.2. Model implementation and results

To complete the model, we use minimally informative priors for the inverse of the three variance parameters ($1/\sigma_r^2, 1/\sigma_s^2, 1/\sigma_q^2 \sim \text{Gamma}(0.5, 0.005)$). This prior has mean 100 and variance 20,000 so it is still a flat prior. Unlike conventional Gamma priors with mean 1, the larger mean of 100 can avoid the chains getting stuck at very low values. Also, we use flat priors $N(0, 100)$ for $\log(\beta)$, $\log(\alpha)$ and $\log(\nu)$ (note the large standard deviation). The model was written in the BUGS language, but implemented in the Nimble package within R (de Valpine et al., 2017).

The run-time using Nimble was 20 min on a 16GB RAM laptop with an i7-8550U CPU. Three MCMC chains were ran for 300,000 iterations, and 200,000 of those were discarded as the burn-in. Only one in ten samples were collected to improve mixing, resulting in a total of 30,000 samples. Here the maximum Gelman-Rubin statistic (across 94 quantities) was 1.03, implying the chains had likely converged to the posterior.

The median estimates of the transmission parameters and the $R_0$ from this case study are within what is expected for this infectious disease (i.e. Salmonella is an agent that mainly spreads via the faecal-oral route). These estimates are also comparable with estimates from other simulation studies (Hill et al., 2008; Lurette et al., 2008) and very similar to what is known from experimental and field studies (Fravalo et al., 2007; Nicholson et al., 2005). The estimate of the transition rate ($\beta$) is slightly higher than that reported in (Lurette et al. (2008)). For a more detailed discussion please see (Correia-Gomes et al., 2014).

The top three panels of Fig. 4 show the posterior distributions of the three transmission parameters $\beta, \alpha$ and $\nu$. These are the parameters of interest, and point estimates of these can be used to run SIR models, noting that the associated uncertainty can also be propagated by using...
the MCMC samples. The bottom left panel shows the posterior density of \( R_0 = \beta/\alpha \), the basic reproduction ratio which quantifies the number of secondary cases to which a primary case gives rise during the infectious period. If \( R_0 < 1 \) then the disease is receding, but \( R_0 > 1 \) implies the disease is spreading. The bottom middle panel shows the posterior mean and 95% credible intervals of \( R_1 \) given the data. This is the temporal random effect that captures any latent temporally varying effects in the transition from susceptible to infectious, in cohort 1. Lastly, the bottom right plot shows the posterior density of the probability of not detecting an infected case, \( \pi_{ND} \), showing that although it is small, it is non-zero. As expected both tests are imperfect as bacteriology lacks sensitivity given intermittent shedding of Salmonella by infected pigs, and there is a delay between infection and expression of antibodies detected by serology. It was consequently important to include test sensitivity within the model as not accounting for this lack of sensitivity could generate a lower transmission rate (Correia-Gomes et al., 2014).

2.3.3. Comments as a first time user

Building the model in WinBUGS was relatively straightforward, existing example code from different models were easily replicable, which made the process behind building the model easier. We have started with a simpler model (without cohort effects) and then step-by-step incorporated additional complexity. At each step the priors and parameters had to be adjusted to prevent the model from crashing. As mentioned previously the WinBUGS trap windows pop up each time an unsatisfactory model is run and deciphering the convoluted error messages can be difficult and time consuming to amend the model.

For complex models applied to large datasets, model implementation can become increasingly slow. However, the option of using BUGS with R for model implementation, and the recent availability of multiBUGS have made BUGS more computationally efficient. The process behind model development and implementation was illustrated using the examples in this article. However, once the software becomes limiting, it is likely that alternative programming languages that enable alternative implementation to MCMC, or are just faster, would need to be considered. Moderate speed is not a issue unique to BUGS, so the choice of programming language becomes a trade-off between mode run-time and user friendliness.

This article aims to illustrate the utility of using BUGS rather than being an exhaustive list of models that can be implemented, but some additional frameworks are worth mentioning. Accounting for correlation between parameters and non-independent data are important for spatiotemporal analyses, for which several methods are available, such as the CAR model that was trialed using the OPV data. This class of auto-regressive models have been used to account for spatial infectious disease data and covariates not being fully independent, and enables a robust analysis (for example see Lawson (2009) and a specific application to dengue modelling by Lowe et al. (2013)). If it is not possible to specify a model using the distributions provided within BUGS, the likelihood function can be specified using the ‘ones’ or ‘zeros’ trick (and the log-likelihood minimised (OpenBUGS, 2006)) and the MCMC machinery used to estimate the posterior distribution (example provided in the SI).

The versatility, ease of use and implementation of Bayesian analysis makes BUGS an excellent tool in infectious disease modelling. For the authors, this means that BUGS has been used for over 10 years and will continue to be used in both teaching and research.

Declaration of Competing Interest

The authors have no conflicts of interest to declare.

Acknowledgements

This work was part-funded funded (MA and KO) by the Bill and Melinda Gates Foundation (OPP1191821) and RL was supported by a Royal Society Dorothy Hodgkin Fellowship. The authors thank the UK National Institute for Health Research Health Protection Research Unit (NIHR HPRU) in Modelling Methodology at Imperial College London in partnership with Public Health England (PHE) for funding (grant HPRU-2012–10080) that enabled us to participate in a seminar series at Imperial College London that was associated with this special edition. The authors thank the reviewers for constructive comments that improved the manuscript.

The full code for each example is provided at https://github.com/kath-o-reilly/Command-BUGS

Appendix A. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.epidem.2019.100361.

References


LSHTM, 2019. Epidemiology MSc at the LSHTM. URL https://www.lshtm.ac.uk/study/
new students/starting-your-course-london/masters-orientation-and-programme-
information/medical-statistics.

ICI, 2019. Epidemiology MSc at Imperial College London. URL https://www.imperial.ac.
.uk.study/pg.medicine/epidemiology/.

Seaman III, J.W., Seaman Jr., J.W., Stamey, J.D., 2012. Hidden dangers of specifying non-

Toft, N., Innocent, G.T., Gettinby, G., Reid, S.W.J., 2007. Assessing the convergence of
Markov Chain Monte Carlo methods: an example from evaluation of diagnostic tests
1016/j.prevetmed.2007.01.003.

Plummer, M., Best, N., Cowles, K., Vines, K., 2006. CODAC convergence diagnosis and
output analysis for mcmc. R News 6, 7–11. URL: https://journal.r-project.org/
archive/.


Tangermann, R.H., Lamoureux, C., Tallis, G., Goel, A., 2017. The critical role of acute
flaccid paralysis surveillance in the Global Polio Eradication Initiative. Int. Health 9,
156–163.

O’Reilly, K.M., et al., 2017. An assessment of the geographical risks of wild and vaccine-
derived poliomyelitis outbreaks in Africa and Asia. BMC Infect. Dis. 17.

Tegegne, A.A., et al., 2018. Characteristics of wild polio virus outbreak investigation and
response in Ethiopia in 2013-2014: implications for prevention of outbreaks due to
importations. BMC Infect. Dis. 18.

WHO, 2019. Immunization Schedules by Country. URL: http://apps.who.int/im-
umination_monitoring/globalsummary/schedules?
schedule=AFRO&
country=SD&%5B%5D=AFRO&
country=SD&%5B%5D=COD&%5B%5D=ECU&%5B%5D=OPV&%5B%5D=OK.

Cutts, F.T., Izueta, H.S., Rhoda, D.A., 2013. Measuring Coverage in MNCH: Design, Im-
plementation, and Interpretation Challenges Associated with Tracking Vaccination