EpiBayes 2-Level Models Vignette

Matt Branan

Updated: June 23, 2015

1 Setting up the Models in R

Run this code once to load the proper packages.

```
library(epiR)  # For the BetaBuster function
library(compiler)  # To compile the larger functions for computational speed
library(coda)  # For processing Bayesian model output
library(shape)  # For nice colorbar legends
library(scales)  # For transparent colors
```

Next, we need to make sure that our package is installed and that the models we will be using are loaded into R's working memory. The file `EpiBayes_0.0.1.tar.gz` can be found in our shared Dropbox folder under `ref` → `Matt Documents`. You will need to make sure that you point R to the actual file using the `setwd` function. For example, if I knew that the file containing the models was stored somewhere on my Mac (using OSX 10.6.8), then I would make sure that the R working directory pointed to the folder containing our file and execute:

```
install.packages("EpiBayes_0.0.1", type = "source", repos = NULL)  # Make sure the
version is correct and the working directory is pointed to where the .tar.gz
file is stored
library(EpiBayes)  # Load the package
```

Before we actually get our hands dirty with the models, we will visit each briefly to get ourselves familiar with the arguments of each and the potential uses of the models.

2 Argument Descriptions and Model Uses

We have code for two models:

a. `EpiBayes_ns`: the hierarchical Bayesian model with n(o) s(torage). This model stores only what is necessary to investigate the simulation results and posterior inference on cluster-level prevalences. No other parameters are tracked and stored in the MCMC chain and so no diagnostics can be investigated. This model is faster than model (b)

b. `EpiBayes_s`: the hierarchical Bayesian model with s(torage). This model stores all realizations of all parameters listed in Section 2.2. Diagnostics can be checked with this model, though it is slower to execute.

Other than the storage differences and the implied differences in computational speed, there are no other differences between the two models. Both are designed to implement the models described in *EpiBayes Proposal (3-level)* in our Dropbox folder and is constructed in the spirit of Branscum et al. (2006) [1].

These models are Bayesian hierarchical models that can serve two main purposes:

- Simulation model: can simulate data under user-specified conditions and run replicated data sets under the Bayesian model to observe the behavior of the system under random realizations of simulated data.
- Posterior inference model: can use actual, observed data from the field, run it through the Bayesian model, and make inference on parameter(s) of interest using the posterior distribution(s).

The outputs of interest if the model is used as a simulation model can be found in Table 1.

Table 1: Outputs from either the `EpiBayes_ns` or `EpiBayes_s` models if used as simulation-type models. These are, generally, frequencies of various behaviors of cluster-level prevalences among replicated data sets.

Statistic	Meaning	Short Meaning
`p2.tilde`	Proportion of replicated data sets that result in probabilities of seeing cluster-level prevalences below `tau.thresh` with `p1` probability	Probability of non-detection or disease-freedom
`p4.tilde`	Proportion of replicated data sets that result in probabilities of seeing cluster-level prevalences above `tau.thresh` with `p1` probability	Probability of detection or disease
`p6.tilde`	Proportion of replicated data sets that result in probabilities of seeing cluster-level prevalences in the interval (`tau.lb`, `tau.ub`) with `p1` probability	Probability of prevalence falling in the interval

In the case of using he model as either a simulation or a posterior inference model, we might wish to investigate the posterior distributions of various parameters returned by the functions implementing our models. In order to manipulate these posterior distributions, we make use of the `coda` package in R, which is a standard package for post-processing of MCMC output.

Both of the models deal with the same set of variables, which can be found in Table 2.

In addition, both models also have the same arguments. These are described in Table 3. We include this to give the user a single place to reference the computational attributes of the arguments as well as English descriptions of the inputs.

Next, we will delve into the output of models (a) and (b) separately (though there will be some overlap).

2.1 No Storage Model

The so-called no storage model, `EpiBayes_ns`, has the arguments as described in Table 3.

The outputs that we can garner from the no storage model are just the bare bones needed to interpret the results in terms of the problem and does not include much in the way of output that can be used to rigorously check for model fit or MCMC convergence properties. From this model, we get the simulation results from Table 1 and the posterior distribution of the cluster-level prevalence. We include the attributes of these outputs in Table 4.

Table 2: Variables involved in both models `EpiBayes_ns` and `EpiBayes_s`. The naming convention "*mat" (where * is a wildcard) is for consistency between models since most of these variables truly are stored as matrices in the storage model, `EpiBayes_s`.

Variable	(3-level) Description	(2-level) Description
`omegamat`	Probability of disease being in the region	Not used
`gammat`	Subzone-level (between subzone) prevalence	Probability of disease being in the region
`z.gammat`	Subzone-level (between subzone) prevalence latent indicator variable	Not used
`taumat`	Cluster-level (between-cluster) prevalence	Same as (3-level)
`z.taumat`	Cluster-level (between-cluster) prevalence latent indicator variable	Same as (3-level)
`pimat`	Subject-level (within-cluster) prevalence	Same as (3-level)
`z.pimat`	Subject-level (within-cluster) prevalence latent indicator variable	Same as (3-level)
`mumat`	Mean prevalence among infected clusters	Same as (3-level)
`psimat`	(Related to) variability of prevalence among infected clusters (inversely related so higher psi $\rightarrow$ lower variance of prevalences among diseased clusters)	Same as (3-level)
`etamat`	Diagnostic test sensitivity	Same as (3-level)
`thetamat`	Diagnostic test specificity	Same as (3-level)
`c1mat`	Latent count of true positive diagnostic test results	Same as (3-level)
`c2mat`	Latent count of true negative diagnostic test results	Same as (3-level)

2.2 Storage Model

The storage model, `EpiBayes_s`, has the arguments as described in Table 3. The outputs from the model are more comprehensive and posterior distributions can be referenced for all variables listed in Table 2. We include the attributes of these matrices output by the storage model for easy reference when manipulating the output in practice. Also in this model, we can access the simulation-based output as before.

2.3 Brief Look at Posterior Distributions

Note, one needs to be careful about the size of each of the arrays you are calling. The last index of any of the variables from above is the MCMC replications and so we would typically always omit the last index when looking at any particular variable. Let's take two examples, one looking at the cluster-level prevalence and the other looking at subject-level prevalences.

If we want to look at the posterior distribution of the cluster-level prevalence (`taumat`) for the first replication, we will note that `taumat` is a matrix with rows indexed by replication and columns by MCMC replications. Then, we will type something like

```
name_of_your_model$taumat[1, 1, ]
```

in order to visually inspect the posterior distribution in the form of a vector. For the second replication, we can type

```
name_of_your_model$taumat[2, 1, ]
```

and so forth. Then, we can make histograms of these distributions if we so desire by the following code:

```
hist(name_of_your_model$taumat[1, 1, ], col = "cyan");box("plot")
```

To observe a trace plot, we can type:

```
plot(name_of_your_model$taumat[1, 1, ], type = "l")
```

for all of the MCMC replications and we can look at the trace plot after a burn-in of 1000 iterations by typing:

```
plot(name_of_your_model$taumat[1, 1, -c(1:1000)], type = "l")
```

If we want to look at the posterior distribution for the subject-level prevalence (`pi`) for the tenth replication in the third cluster, we would type

```
name_of_your_model$pimat[10, 1, ]
```

since the matrix containing the posterior distributions for the subject-level prevalences are indexed by replications in the first dimension, clusters in the second, and MCMC replications in the third. We can make histograms and trace plots using the same code as from the example code involving `taumat`.

Table 3: Arguments for the function `HM_2ns` in the order in which they appear by default. We also include the attributes of each of the arguments so the user has a single place to reference when the type of input is questioned in practice.

Argument	Attributes	Description
`H`	integer scalar	number of subzones/states/HUC's
`k`	integer vector (`H` $\times$ 1)	number of clusters / farms / ponds / herds
`n`	integer vector (`sum(k)` $\times$ 1)	number of subjects / animals / mussels / pigs per cluster (can differ among clusters)
`seasons`	integer vector (`sum(k)` $\times$ 1)	numeric season for each cluster in the order: Summer (1), Fall (2), Winter (3), Spring (4)
`reps`	integer scalar	number of (simulated) replicated data sets
`MCMCreps`	integer scalar	number of iterations in the MCMC chain per replicated data set
`poi`	character scalar	p(arameter) o(f) i(nterest) specifies one of the subzone-level prevalence (`gam`) or the cluster-level prevalence (`tau`), indicating which variable with which to compute the simulation output `p2.tilde`, `p4.tilde`, and `p6.tilde`
`y`	integer matrix (`reps` $\times$ `sum(k)`	an optional input of sums of positive diagnostic testing results if one has a specific set of diagnostic testing outcomes for every subject (will simulate these if this is left as `NULL`)
`mumodes`	real matrix (4×2)	modes and (a) 95th percentiles for mode <= 0.50 or (b) 5th percentiles for mode > 0.5 for season-specific mean prevalences for diseased clusters in the order: Summer, Fall, Winter, Spring
`poi.thresh`	real scalar	Threshold that we must show `poi` prevalence is below to declare disease freedom
`tau.T`	real scalar	assumed true cluster-level prevalence (used to simulate data to feed into the Bayesian model)
`poi.lb` / `poi.ub`	real scalars	lower and upper bounds for posterior `poi` prevalences to show ability to capture `poi` with certain probability
`p1`	real scalar	probability we must show prevalence is below / above the threshold `tau.thresh` or within specified bounds
`psi`	real scalar	(inversely related to) the variability of the subject-level prevalences in diseased clusters
`*parm(*, *)`	real vector (2×1)	the rest of the model inputs in the form *parm(*, *) are the prior parameters for variable * where * can be one of: `omega`, `gam`, `tau`, `eta`, `theta`
`burnin`	integer scalar	number of MCMC iterations to discard from the beginning of the chain

Table 4: Output values from the no storage model and their attributes. Mainly important for reference of the size of the `taumat` matrix to make investigation more straightforward when calling `coda` functions for clean post-processing.

Output Variable	Attributes
`p2.tilde`	real scalar
`p4.tilde`	real scalar
`p6.tilde`	real scalar
`taumat`	real matrix (`reps` $\times$ H $\times$ `MCMCreps`)

Table 5: Output values from the no storage model and their attributes. Mainly important for reference of the size of the `taumat` matrix to make investigation more straightforward when calling `coda` functions for clean post-processing.

Output Variable	Attributes
`p2.tilde`	real scalar
`p4.tilde`	real scalar
`p6.tilde`	real scalar
`taumat`	real matrix (`reps` $\times$ H $\times$ `MCMCreps`)
`gammat`	real matrix (`reps` $\times$ `MCMCreps`)
`omegamat`	real matrix (`reps` $\times$ `MCMCreps`)
`z.gammat`	real matrix (`reps` $\times$ `MCMCreps`)
`z.taumat`	real matrix (`reps` $\times$ H $\times$ `MCMCreps`)
`pimat`	real array (`reps` $\times$ `sum(k)` $\times$ `MCMCreps`)
`z.pimat`	real array (`reps` $\times$ `sum(k)` $\times$ `MCMCreps`)
`mumat`	real matrix (`reps` $\times$ 4 $\times$ `MCMCreps`)
`psimat`	real scalar
`etamat`	real matrix (`reps` $\times$ `MCMCreps`)
`thetamat`	real matrix (`reps` $\times$ `MCMCreps`)
`c1mat`	real array (`reps` $\times$ `k` $\times$ `MCMCreps`)
`c2mat`	real array (`reps` $\times$ `k` $\times$ `MCMCreps`)
`mumh.tracker`	real matrix (`reps` $\times$ 4)
`y`	real matrix (`reps` $\times$ `sum(k)`)

References

[1] Branscum, A., Johnson, W. and Gardner, I. Sample size calculations for disease freedom and prevalence estimation surveys. *Statistics in Medicine 25* (2006), 2658-2674.