EpiBayes 3-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
library(EpiBayes)  # Load our package
```

Next, we will use the hierarchical Bayesian model to investigate a 3-level sampling design in which we have one region with three subzones of interest. Two subzones have ten farms sampled and we sample 100 cows a piece and the third subzone has fifty clusters sampled with 100 cows sampled a piece. We implement the storage model, `EpiBayes_s`, to investigate the posterior distributions of the cluster-level prevalences after one year of sampling in which we see only the third subzone infected in which we see ten farms with ten infected cows and fifteen farms with fifteen cows infected.

We also use the `EpiBayesHistorical` function and its methods to investigate the ways we may combine several years' information into one statement about the cluster-level prevalence of the disease under investigation.

2 Examples

2.1 Example 1: 3-Level Posterior Inference

Consider visiting a region in which there are three subzones (could be states) of interest. Two of the subzones have ten farms sampled, and we sample 100 cows per farm. The third subzone has fifty sampled farms with 100 cows sampled on each. All of the sampling during this year was done in the Fall. During this season, the average subject-level prevalence of the disease is about 10%. We specify that the disease is somewhere in the region and that we expect about 40% of the subzones to be infected and the sensitivity and specificity of the diagnostic test used is around 90%.

When we go out and sample our cows, we find the first two subzones to have no animals infected by our diagnostic testing procedure, but find ten farms with ten infected cows and fifteen farms with fifteen infected cows in the third subzone.

First, we construct a matrix with a single row that demonstrates the outcomes of our observations. Once the prior distributions have been decided upon, we may call the actual model – we'll be using the storage model in this case just so we can check some of the posterior distributions if we would like to. The function call will look something like the following. We have included annotations next to each argument so that it is clear what each argument is and why it had been initialized as such.

```
obs.y = matrix(c(
                rep(0, 10),  # Subzone 1
                rep(0, 10),  # Subzone 2
                rep(10, 10), rep(15, 15), rep(0, 25)  # Subzone 3
                ),
                nrow = 1
        )
```

```
set.seed(2015)  # To ensure reproducible results
example1.run = EpiBayes_s(
    H = 3,  # 3 subzones
    k = c(rep(10, 2), rep(50, 1)),  # 10 farms in two subzones, 50 in
        # the third subzone
    n = rep(100, 70),  #100 cows sampled in each of the 70 clusters
    seasons = rep(2, 70),  # Seasons corresponding to each cluster
        # (1 for summer, 2 for fall, 3 for winter, 4 for spring)
        mumodes = matrix(c(
                0.10, 0.50,
                0.10, 0.50,
                0.10, 0.50,
                0.10, 0.50
                ), 4, 2, byrow = TRUE
        ), # Modes and 95th percentiles of
        # subject - level prevalences for each season in order
    reps = 1,  # 1 replicated data set in this simulation
    MCMCreps = 100,  # 100 MCMC iterations per replicated data
        # set (increasing this would be a good idea for real data but slows
        # things down a lot)
    poi = "tau",  # Want inference on cluster-level prevalence
    y = obs.y,  # Specify the number of positive test results we saw for each farm
    pi.thresh = 0.05,  # The 5% threshold (design prevalence) for the
        # cluster - level prevalence
    tau.thresh = 0.02,  # The 2% threshold (design prevalence) for the
        # cluster - level prevalence
    gam.thresh = 0.01,  # The 1% threshold (design prevalence) for the
        # cluster - level prevalence
    tau.T = 0.20,  # The "true cluster - level prevalence" that we simulate our
        # data with (this means about 20% of our clusters in each replicated
        # data set will be diseased and will have a truly positive
        # subject - level prevalence)
    poi.lb = 0,  # The lower bound for estimating the cluster - level
        # prevalence (not of interest here)
    poi.ub = 1,  # The upper bound for estimating the cluster - level
        # prevalence (not of interest here)
    p1 = 0.95,  # The probability (used like a confidence) that we must show
        # our cluster - level prevalence is above 2% in order to count that
        # replicated data set as one in which we detected the disease
```

```
    psi = 4,  # The variability of the prevalences among infected clusters within
        # the subzone
    omegaparm = c(1000, 1),  # Prior parameters for omegamat (the probability
        # of the disease being in the region)
    gamparm = c(20, 30),  # Prior parameters for  gammat (the subzone-level
        # prevalence)
    tauparm = c(1, 1),  # Prior parameters for  taumat (the cluster - level
        # prevalence)
    etaparm = c(10, 1),  # Prior parameters for  etamat (the diagnostic
        # test sensitivity)
    thetaparm = c(10, 1),  # Prior parameters for  thetamat (the diagnostic
        # test specificity)
    burnin = 10  # The amount of MCMC iterations to "burn"
    )
```

We can investigate the output using the `summary` and `plot` methods for the output object type just like we had in the 2-level vignette examples. The only difference here is that we have more parameters to investigate (specifically, more subject-level and cluster-level prevalences and a new subzone-level prevalence to observe). Here, since we have only one replication (the supplied observed data in the `obs.y` matrix) then we don't really need to concern ourselves with the simulation output values from the summary output.

```
## Summary
example1.sum = summary(example1.run)

## Simulation output for parameter of interest (poi)
##  *p2.tilde: Percentage of the time the disease is not detected above the disease threshold
##  *p4.tilde: Percentage of the time the disease is detected above the disease threshold
##  *p6.tilde: Percentage of the time the disease is detected between the user-supplied lower a
##
##  p2.tilde  p4.tilde p6.tilde
##         0 0.3333333        1
##
##  ------------------------------------------------------------------------------
## gam: Subzone-level prevalence
##           Mean         SD   Naive SE Time-series SE Lower HPD Limit
## [1,] 0.4093176 0.06443988 0.00679256    0.004670392       0.2918837
##      Upper HPD Limit
## [1,]       0.5132293
##
##  ------------------------------------------------------------------------------
## tau 1: Cluster-level prevalence in subzone 1
##            Mean         SD    Naive SE Time-series SE Lower HPD Limit
## [1,] 0.01267463 0.02252947 0.002374815    0.003387934    9.415726e-05
##      Upper HPD Limit
## [1,]      0.05600567
##
##  ------------------------------------------------------------------------------
```

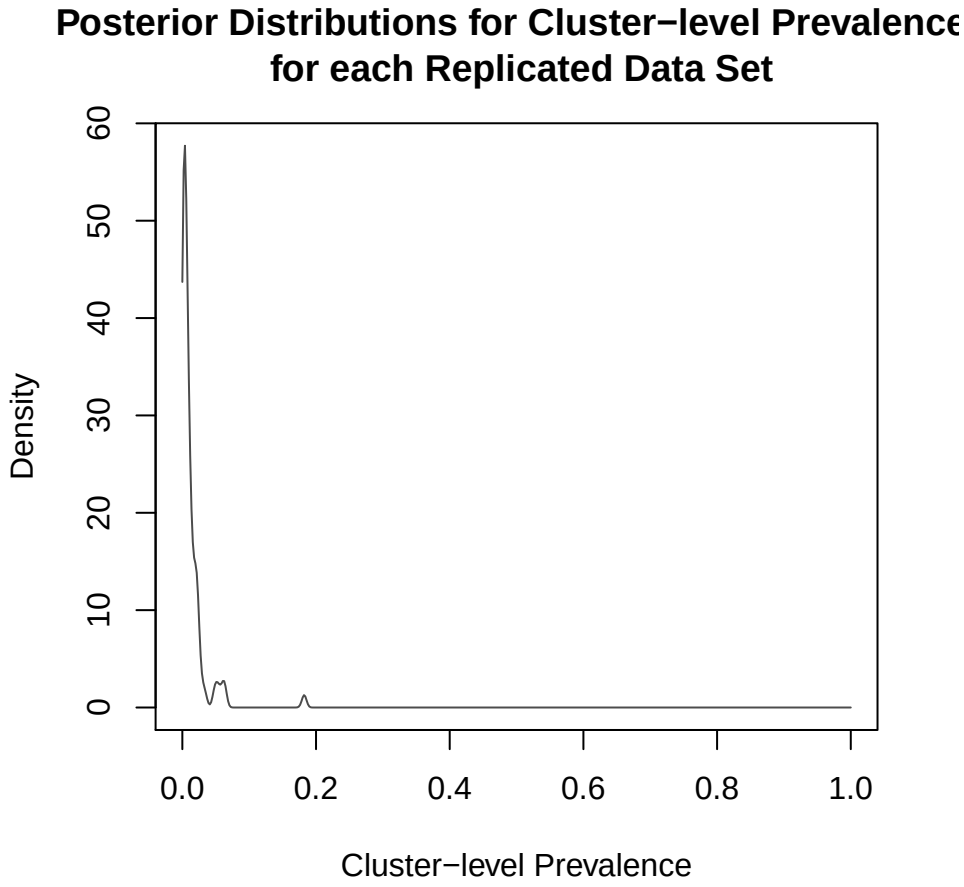
```
## tau 2: Cluster-level prevalence in subzone 2
##            Mean         SD    Naive SE Time-series SE Lower HPD Limit
## [1,] 0.01847827 0.03815366 0.004021749    0.004733938    8.162337e-06
##      Upper HPD Limit
## [1,]      0.06306321
##
##  ------------------------------------------------------------------------
## tau 3: Cluster-level prevalence in subzone 3
##           Mean         SD    Naive SE Time-series SE Lower HPD Limit
## [1,] 0.5025969 0.05950448 0.006272323    0.006272323       0.3772608
##      Upper HPD Limit
## [1,]       0.6160542
##
##  ------------------------------------------------------------------------
```

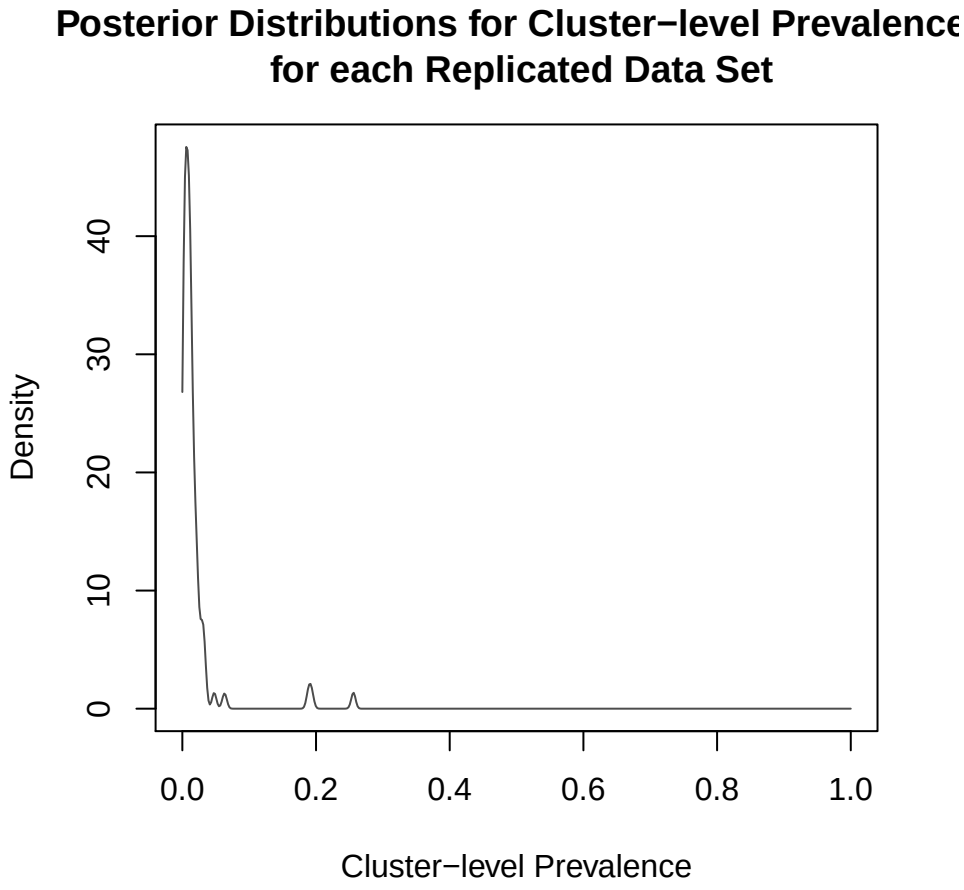
```
example1.sum
```

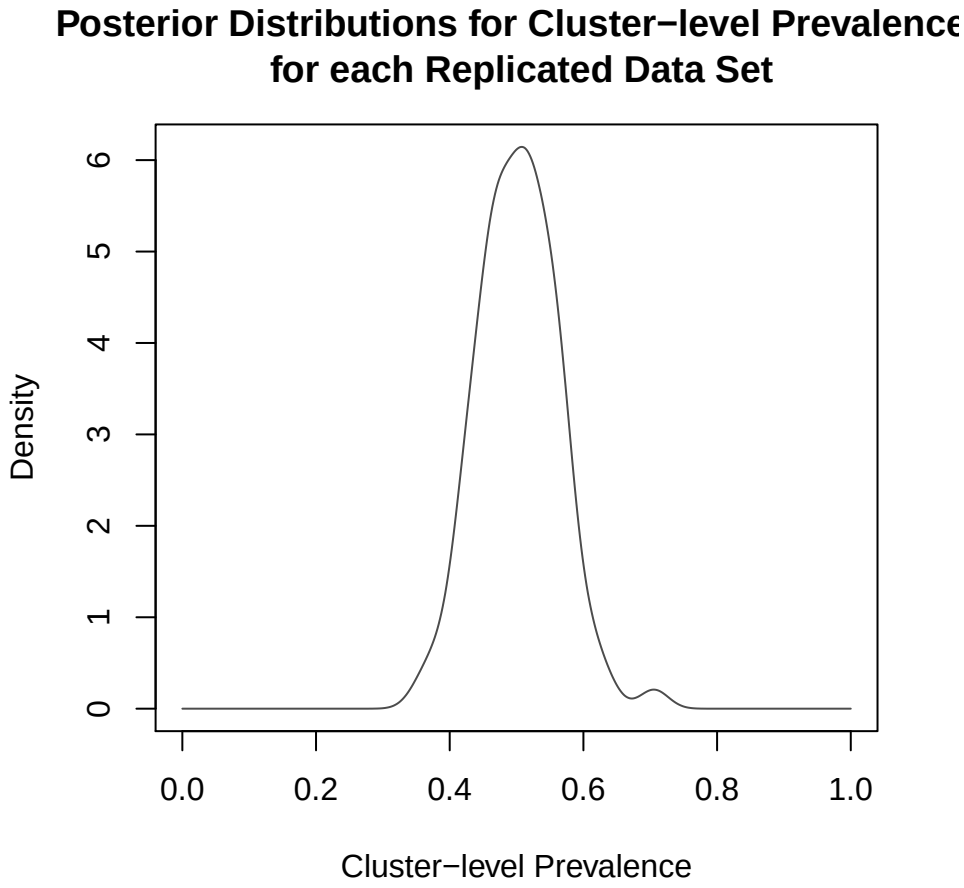
```
## Simulation output for parameter of interest (poi)
##  *p2.tilde: Percentage of the time the disease is not detected above the disease threshold
##  *p4.tilde: Percentage of the time the disease is detected above the disease threshold
##  *p6.tilde: Percentage of the time the disease is detected between the user-supplied lower a
##
##  p2.tilde  p4.tilde p6.tilde
##         0 0.3333333        1
##
##  ------------------------------------------------------------------------
## gam: Subzone-level prevalence
##           Mean         SD   Naive SE Time-series SE Lower HPD Limit
## [1,] 0.4093176 0.06443988 0.00679256    0.004670392       0.2918837
##      Upper HPD Limit
## [1,]       0.5132293
##
##  ------------------------------------------------------------------------
## tau 1: Cluster-level prevalence in subzone 1
##            Mean         SD    Naive SE Time-series SE Lower HPD Limit
## [1,] 0.01267463 0.02252947 0.002374815    0.003387934    9.415726e-05
##      Upper HPD Limit
## [1,]      0.05600567
##
##  ------------------------------------------------------------------------
## tau 2: Cluster-level prevalence in subzone 2
##            Mean         SD    Naive SE Time-series SE Lower HPD Limit
## [1,] 0.01847827 0.03815366 0.004021749    0.004733938    8.162337e-06
##      Upper HPD Limit
## [1,]      0.06306321
##
##  ------------------------------------------------------------------------
## tau 3: Cluster-level prevalence in subzone 3
```

```
##           Mean         SD    Naive SE Time-series SE Lower HPD Limit
## [1,] 0.5025969 0.05950448 0.006272323    0.006272323       0.3772608
##      Upper HPD Limit
## [1,]       0.6160542
##
##  ------------------------------------------------------------------------------
```

```
## Plot the posterior distributions of cluster-level prevalence
plot(example1.run)
```

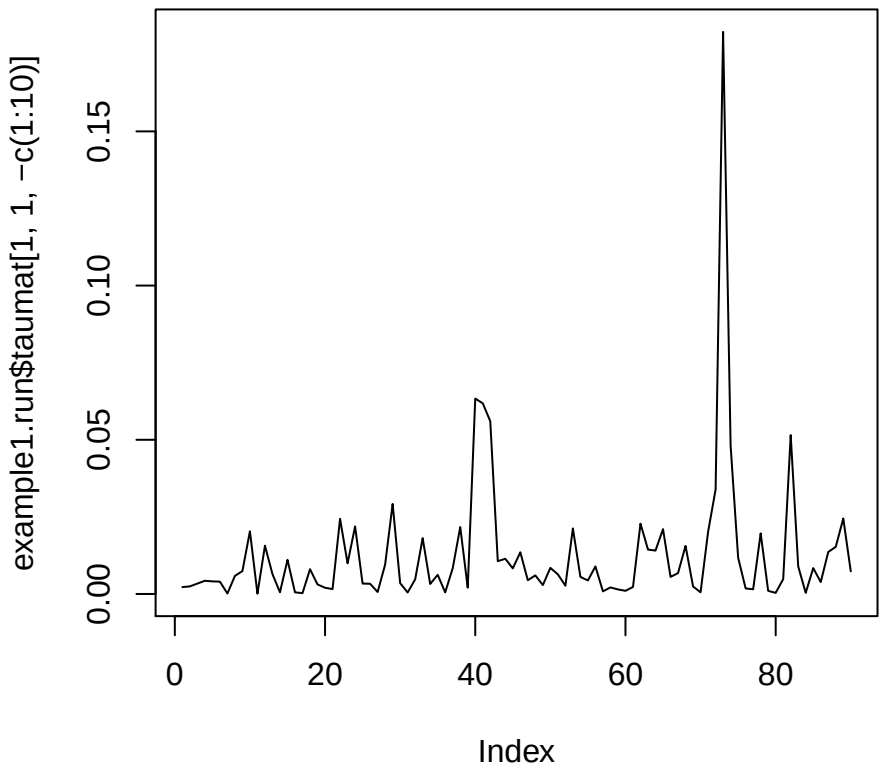



Posterior Distributions for Cluster−level Prevalence
for each Replicated Data Set
Density
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0
Cluster−level Prevalence

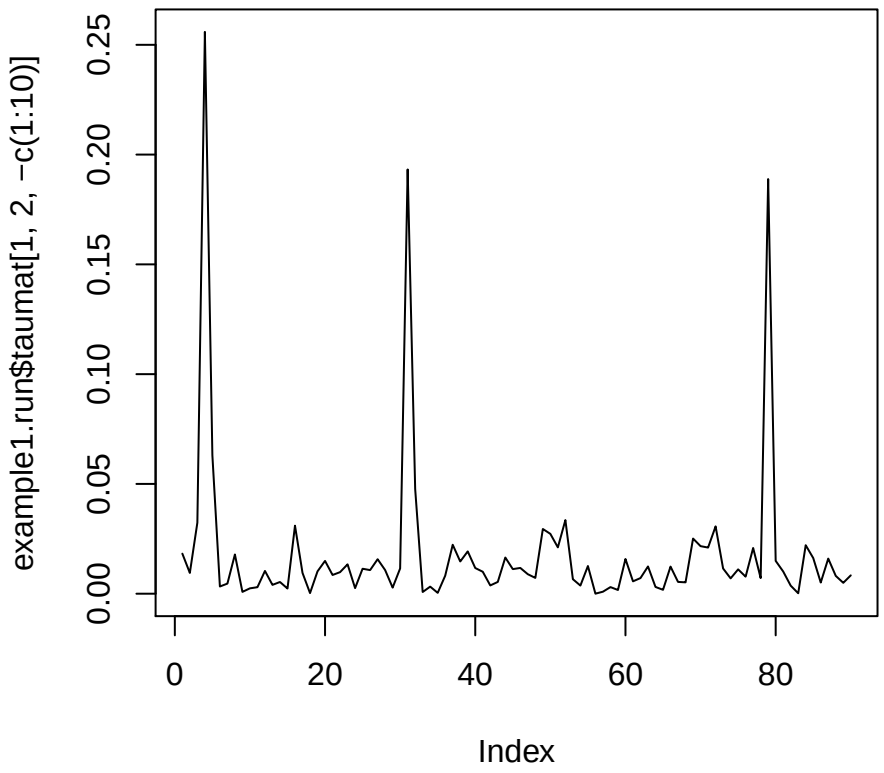


We can also look at some trace plots and posterior distribution density estimates for some of the `taumat` and some of the `pimat` chains. Notice that we have eliminated the burnin iterations that we had defined in the `EpiBayes_s` function call.

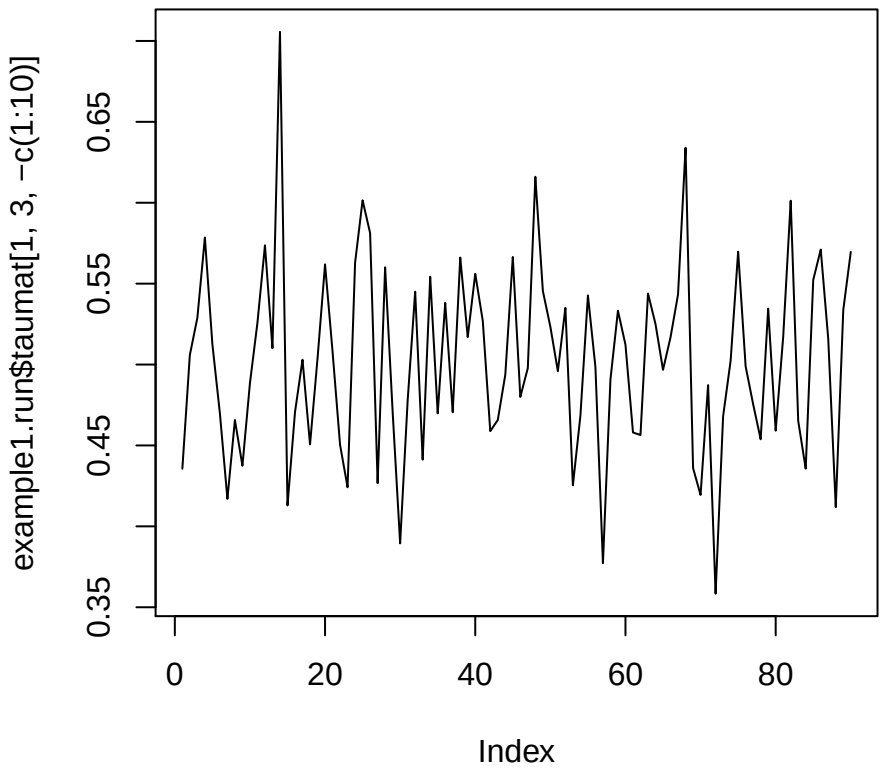
```
## Trace plots

## Tau
        # Tau for the first subzone
        plot(example1.run$taumat[1, 1, -c(1:10)], type = "l")
```



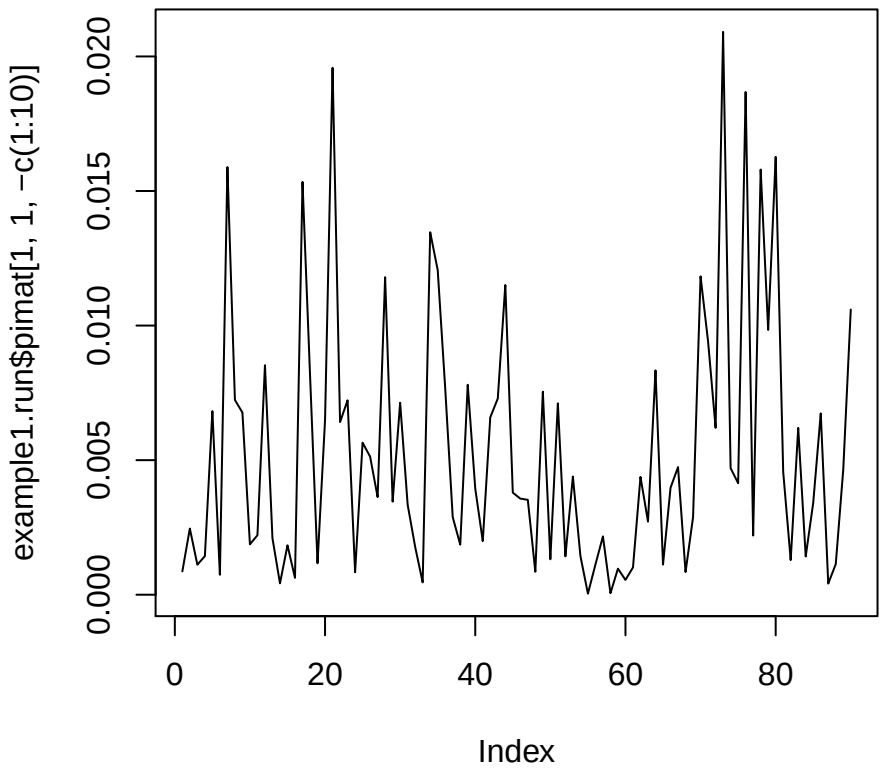

```
# Tau for the second subzone
plot(example1.run$taumat[1, 2, -c(1:10)], type = "l")
```



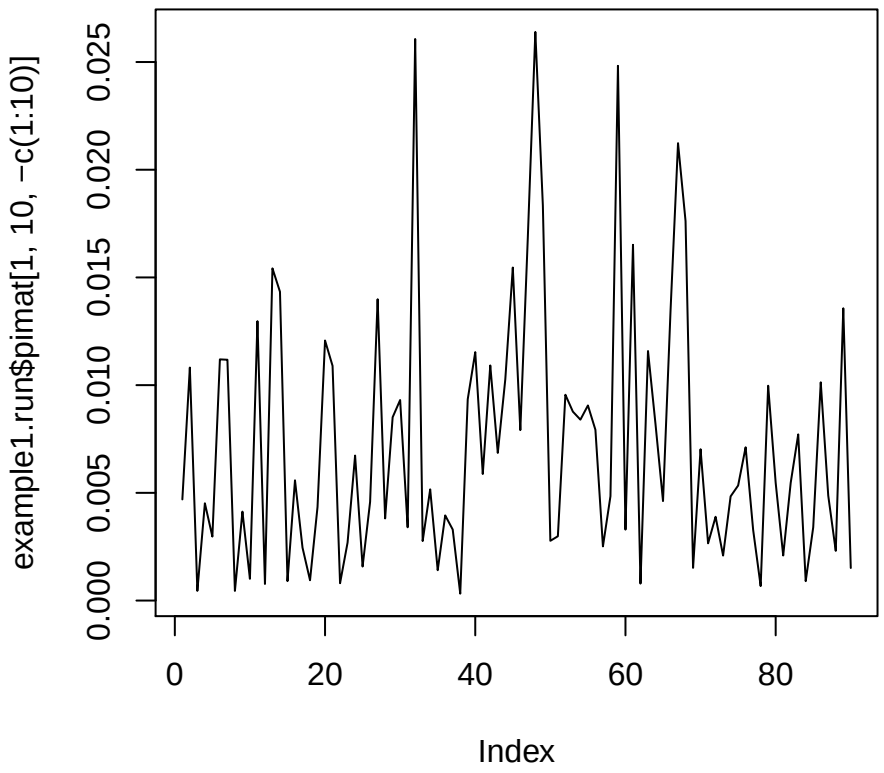

```
# Tau for the third subzone
plot(example1.run$taumat[1, 3, -c(1:10)], type = "l")
```



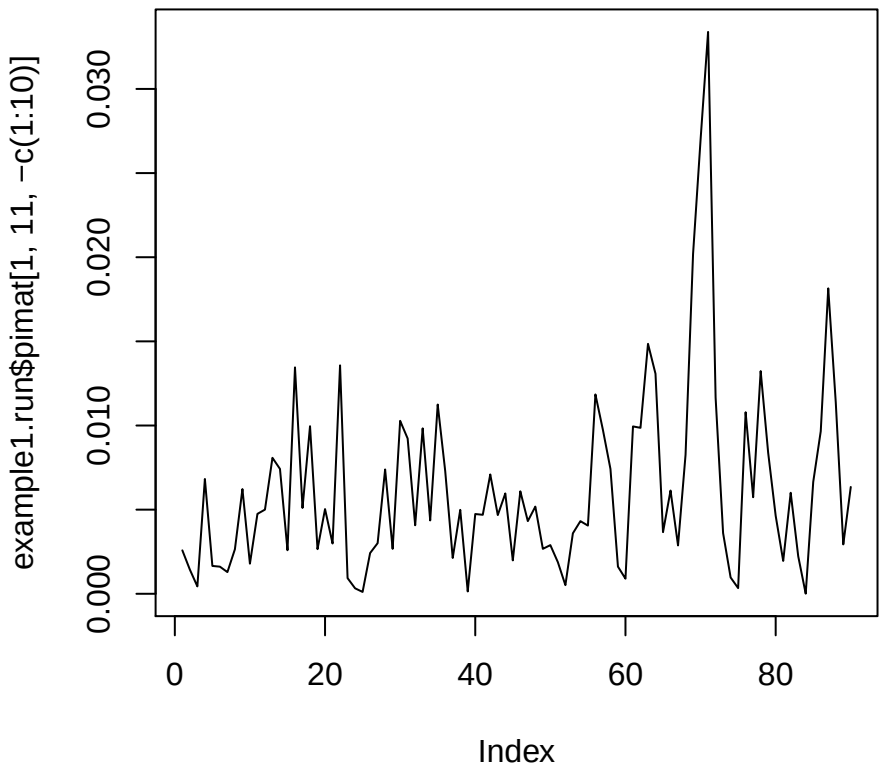

```
## Pi
        # Pi for the first farm in the first subzone
        plot(example1.run$pimat[1, 1, -c(1:10)], type = "l")
```



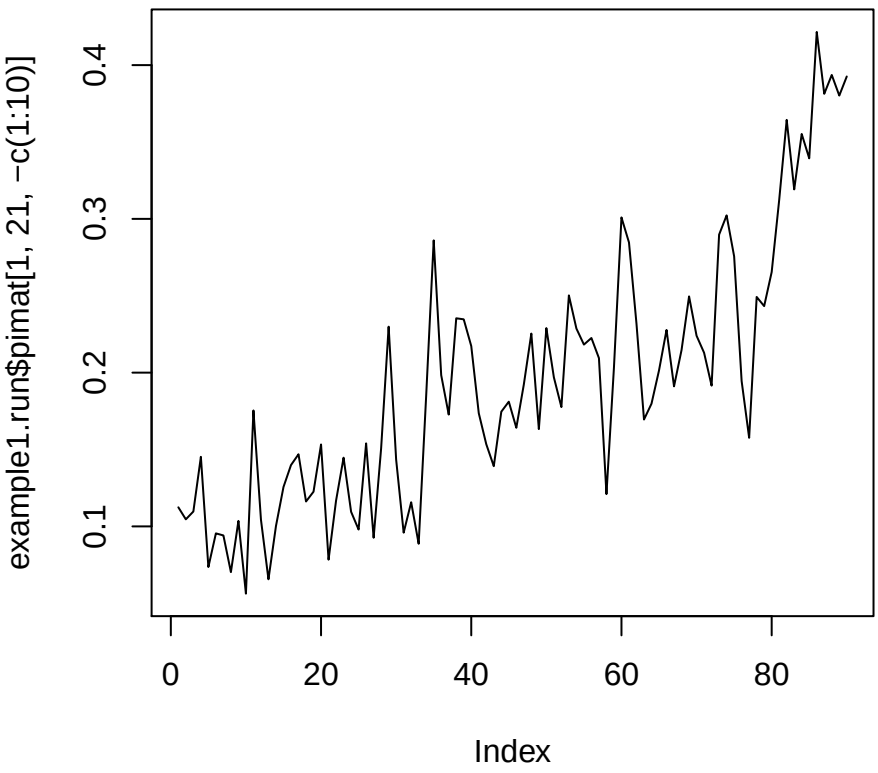

```
# Pi for the tenth farm in the first subzone
plot(example1.run$pimat[1, 10, -c(1:10)], type = "l")
```



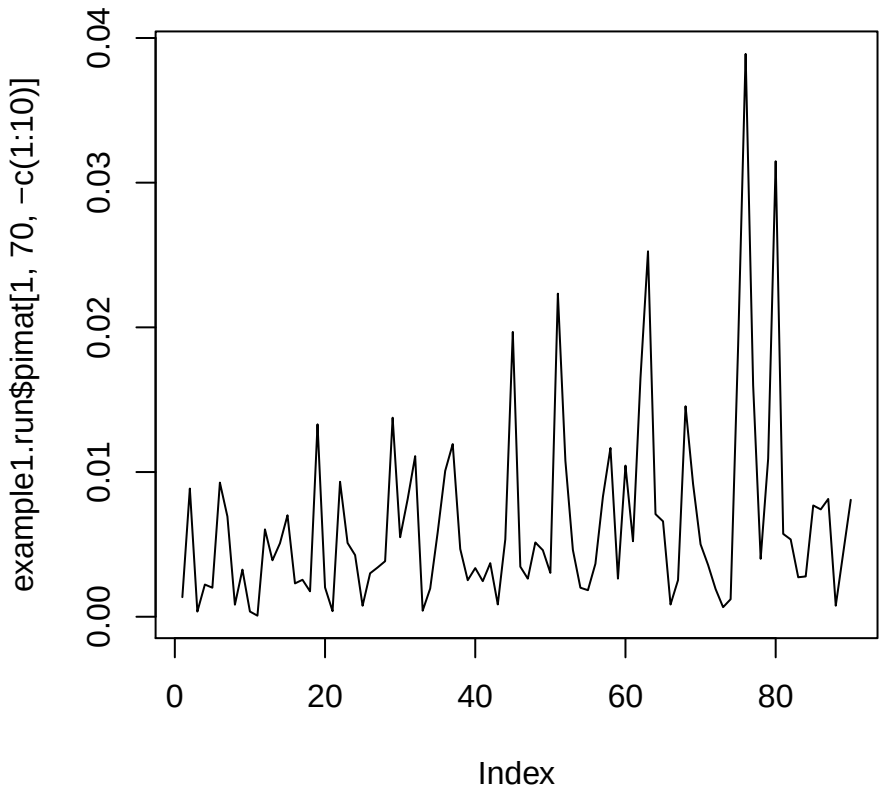

```
# Pi for the first farm in the second subzone
plot(example1.run$pimat[1, 11, -c(1:10)], type = "l")
```



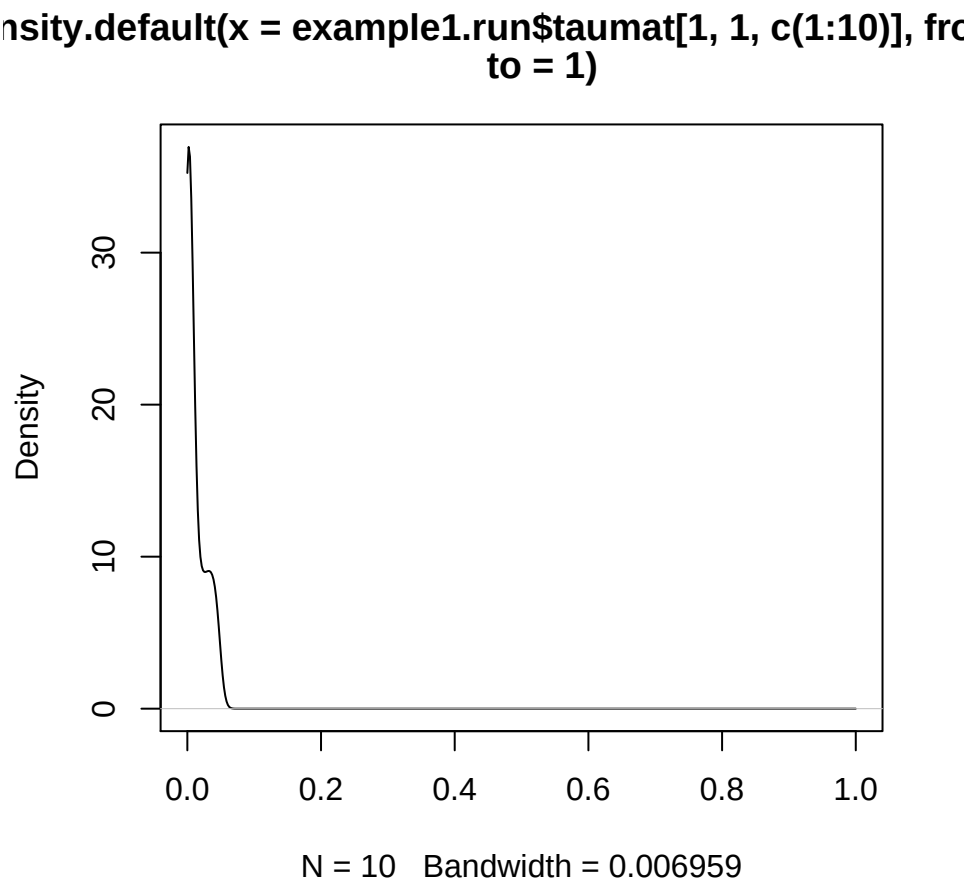

```
# Pi for the first farm in the third subzone
plot(example1.run$pimat[1, 21, -c(1:10)], type = "l")
```



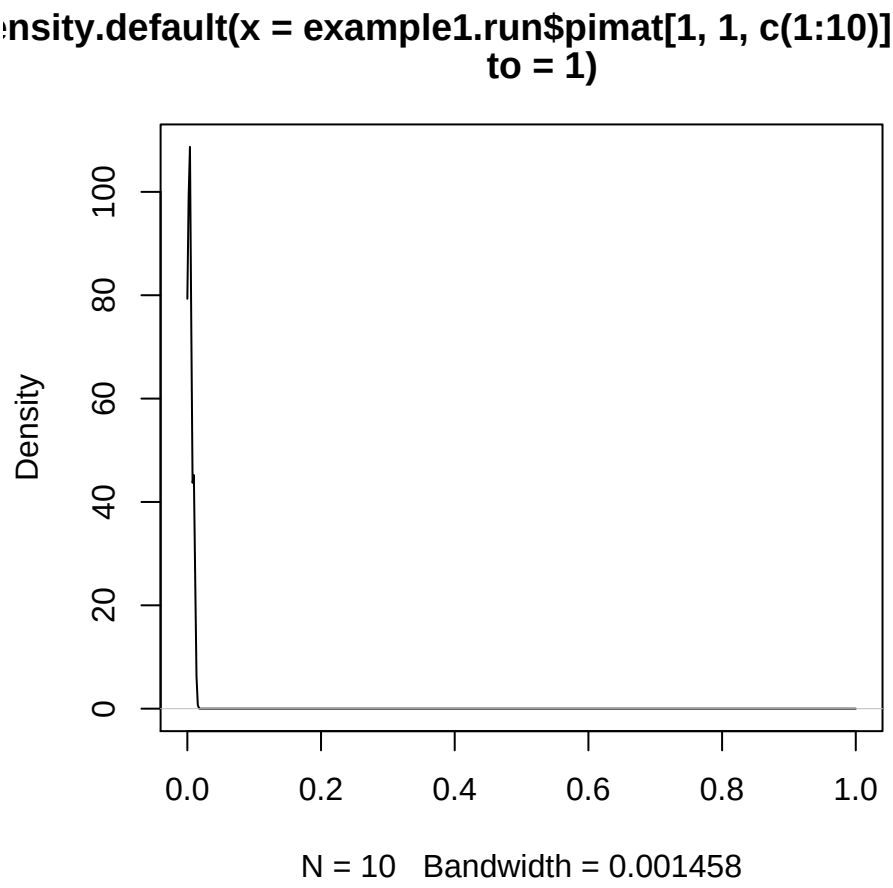

```
# Pi for the fiftieth farm in the third subzone
plot(example1.run$pimat[1, 70, -c(1:10)], type = "l")
```




```
## Histograms

## Tau
        # Tau for the first subzone
        plot(density(example1.run$taumat[1, 1, c(1:10)], from = 0, to = 1))
```




```
## Pi
        # Pi for the first farm in the first subzone
        plot(density(example1.run$pimat[1, 1, c(1:10)], from = 0, to = 1))
```



2.2 Example 2: Historical Updating

Suppose that we have a the same situation as in Example 1 but now we have a different objective. Instead of making posterior inference about the disease prevalences at various levels in the hierarchical sampling procedure, we would like to determine how one may aggregate data across time periods. For example, we could have performed the sampling mentioned in Example 1 in 2010, but we also have sampling data from 2011-2014 as well. Ignoring introduction risk, and using the posterior distribution for the cluster-level prevalence for the prior for the same parameter in the next year and carrying this forward for all of the years of data we have, we can combine our yearly data into an overall statement about the cluster-level prevalence at the end of 2014.

First, we must construct our matrix of observed data. We need to construct a matrix such that every row denotes a cluster and we have columns: Year (or, equivalently, period of collection), Subzone, Cluster size, Season (1-4), and Y (the number of positive diagnostic test results in that cluster).

We already have the 2010 data so we'll just need to 'observe' four more years of data.

```
year = rep(c(2010:2014), each = 70)
subz = rep(rep(c("First", "Second", "Third"), c(10, 10, 50)), 5)
size = rep(100, 70*5)
season = rep(2, 70*5)
```

```
y = matrix(c(
                rep(0, 10), #Year 2010: Subzone 1
                rep(0, 10), #Year 2010: Subzone 2
                rep(10, 10), rep(15, 15), rep(0, 25), #Year 2010: Subzone 3
                rep(2, 10), #Year 2011: Subzone 1
                rep(0, 10), #Year 2011: Subzone 2
                rep(5, 10), rep(10, 15), rep(0, 25), #Year 2011: Subzone 3
                rep(0, 10), #Year 2012: Subzone 1
                rep(4, 10), #Year 2012: Subzone 2
                rep(0, 10), rep(5, 15), rep(0, 25), #Year 2012: Subzone 3
                rep(8, 10), #Year 2013: Subzone 1
                rep(0, 10), #Year 2013: Subzone 2
                rep(0, 10), rep(0, 15), rep(0, 25), #Year 2013: Subzone 3
                rep(4, 10), #Year 2014: Subzone 1
                rep(0, 10), #Year 2014: Subzone 2
                rep(0, 10), rep(0, 15), rep(0, 25) #Year 2014: Subzone 3
                ),
                ncol = 1
        )

example2.inputdf = data.frame(year, subz, size, season, y)
```

```
set.seed(2015)
example2.run = EpiBayesHistorical(
        input.df = example2.inputdf,  # Our input matrix
        orig.tauparm = c(1, 1),  # tau prior parameters in the first year
        burnin = 1,  # Number of MCMC iterations to burn
        MCMCreps = 10,  # Number of MCMC iterations
        tau.T = 0.2,  # Doesn't matter since reps = 1
        poi = "tau",  # Leave parameter of interest as cluster-level prevalence
        mumodes = matrix(c(
                 0.10, 0.50,
                 0.10, 0.50,
                 0.10, 0.50,
                 0.10, 0.50
                 ), 4, 2, byrow = TRUE
         ),# Season-specific average subject-level
                # prevalences in infected clusters
         pi.thresh = 0.05,  # The 5% threshold (design prevalence) for  the
                # cluster - level prevalence
         tau.thresh = 0.02,  # Doesn't matter since reps = 1
         gam.thresh = 0.01,  # Doesn't matter since reps = 1
        poi.lb = 0,  # Doesn't matter since reps = 1
        poi.ub = 1,  # Doesn't matter since reps = 1
        p1 = 0.95,  # Doesn't matter since reps = 1
        psi = 4,  # (related to) variability of subject-level prevalences in
```

```
                # infected clusters
        omegaparm = c(1000, 1),  # Prior parameters for the probability of the
                # disease being in the region (almost always 1)
        gamparm = c(20, 30),  # Prior parameters for the subzone-level prevalence
                # (mean of about 0.4)
        etaparm = c(10, 1),  # Prior parameters for diagnostic test sensitivity
                # (mean of about 0.9)
        thetaparm = c(10, 1)  # Prior parameters for diagnostic test specificity
                # (mean of about 0.9)
        )
```

We may observe the behavior of the posterior cluster-level prevalence distributions across years and for each subzone (each subzone gets its own plotting window) using the `plot` method for the historical function output.

```
## Plot the posterior distributions of cluster-level prevalence with
        # one plotting window for each subzone and colors ranging
        # across years
plot(example2.run)
```

We can also summarize the historical posterior distributions by observing the posterior means, quantiles, or variances for each subzone and track those summary statistics throughout the years. We can take those summaries and plot them as well.

```
## Summaries
        # By mean
        example2.meansum = summary(example2.run, sumstat = "mean",
                time.labels = 2010:2014)

## Matrix of posterior means of cluster-level prevalence across 5 time periods
##
##               2010       2011      2012       2013       2014
## First  0.012865964 0.02093056 0.0144963 0.24695356 0.14846791
## Second 0.007249445 0.02720466 0.2069557 0.03057850 0.03612942
## Third  0.265926862 0.18996917 0.1798568 0.08220457 0.07282204

        example2.meansum

## Matrix of posterior means of cluster-level prevalence across 5 time periods
##
##               2010       2011      2012       2013       2014
## First  0.012865964 0.02093056 0.0144963 0.24695356 0.14846791
## Second 0.007249445 0.02720466 0.2069557 0.03057850 0.03612942
## Third  0.265926862 0.18996917 0.1798568 0.08220457 0.07282204

        # By 95th percentiles
        ## Summaries
        example2.95persum = summary(example2.run, sumstat = "quantile",
                prob = 0.95, time.labels = 2010:2014)
```

```
## Matrix of posterior quantiles of cluster-level prevalence across 5 time periods
##
##              2010      2011       2012       2013      2014
## First  0.03045150 0.0716979 0.05492878 0.40380106 0.3264984
## Second 0.01885031 0.1056588 0.42883067 0.09313852 0.1020940
## Third  0.46333213 0.3660815 0.47370083 0.26115530 0.3669097
```

```
        example2.95persum
```

```
## Matrix of posterior quantiles of cluster-level prevalence across 5 time periods
##
##              2010      2011       2012       2013      2014
## First  0.03045150 0.0716979 0.05492878 0.40380106 0.3264984
## Second 0.01885031 0.1056588 0.42883067 0.09313852 0.1020940
## Third  0.46333213 0.3660815 0.47370083 0.26115530 0.3669097
```

```
## Plotting the summaries across time
        # Plot means
        plot(example2.meansum)
        # Can add a line to compare to a certain design prevalence
        abline(h = 0.05, lty = 2, col = "black", lwd = 2)

        # Plot 95th percentiles
        plot(example2.95persum)
        # Can add a line to compare to a certain design prevalence
        abline(h = 0.05, lty = 2, col = "black", lwd = 2)
```

Note: The above examples are not meant to reflect reality. Notice that the `MCMCreps` in both examples are set very low. This would have been bad if executed in practice, but was set so in order to ensure quick build times for this vignette.