Journal of Statistical Software

November 2014, Volume 61, Issue 13. *http://www.jstatsoft.org/*

BayesLCA: An R Package for Bayesian Latent Class Analysis

Arthur White
University College Dublin

Thomas Brendan Murphy
University College Dublin

Abstract

The **BayesLCA** package for R provides tools for performing latent class analysis within a Bayesian setting. Three methods for fitting the model are provided, incorporating an expectation-maximization algorithm, Gibbs sampling and a variational Bayes approximation. The article briefly outlines the methodology behind each of these techniques and discusses some of the technical difficulties associated with them. Methods to remedy these problems are also described. Visualization methods for each of these techniques are included, as well as criteria to aid model selection.

Keywords: latent class analysis, EM algorithm, Gibbs sampling, variational Bayes, model-based clustering, R.

1. Introduction

Populations of interest can often be divided into homogenous subgroups, although such groupings may never be explicitly observed. Commonly, it is of interest both to identify such divisions in a population of interest and succinctly describe their behavior. Latent class analysis (LCA) is a type of model-based clustering which concerns the study of binary data drawn from such populations. Examples of such data can include the correct or incorrect answers submitted during an exam (Bartholomew and Knott 1999), the symptoms presented by persons with major depressive disorder (Garrett and Zeger 2000), or a disability index recorded by long-term survey (Erosheva, Fienberg, and Joutard 2007).

While the R (R Core Team 2014) environment for statistical computing features several packages for finite mixture models, many are concerned with continuous data, such as mixtures of multivariate Gaussian distributions (**mclust**, Fraley and Raftery 2007), regression models (**flexmix**, Leisch 2004), or some combination therein (**mixtools**, Benaglia, Chauveau, Hunter, and Young 2009). While functions to perform LCA are available in packages such as **e1071**

(Dimitriadou, Hornik, Leisch, Meyer, and Weingessel 2014) and in particular **poLCA** (Linzer and Lewis 2011), these limit the user to performing inference within a maximum likelihood estimate, frequentist framework. No dedicated package for performing LCA within a Bayesian paradigm yet exists.

The main aims of the **BayesLCA** package are:

- Cluster observations into groups.
- Perform inference on the model posterior. This may be done by obtaining *maximum a posteriori* (MAP) and posterior standard deviation estimates; iteratively sampling from the posterior; or by approximating the posterior with another distribution of a convenient form.
- Report both parameter estimates and posterior standard deviations.
- Provide plotting tools to visualize parameter behavior and assess model performance.
- Provide summary tools to help assess model selection and fit.

Users of the package may also wish to include prior information into their analysis, something outside the scope of frequentist analysis. There may be multiple reasons for doing so: expert information may be available; certain boundary solutions may be viewed as unrealistic, and as such to be avoided; differing prior information may be supplied to the model in order to check the robustness of results. However, in this paper we do not dwell on this issue, instead emphasizing the methods for parameter estimation and visualization which are available in the package.

Note that while the package emphasizes inference within a Bayesian framework, inference may still be performed from a frequentist viewpoint. For example, the use of the expectation maximization (EM) algorithm, together with the specification of (proper) uniform priors for all model parameters, is the equivalent of obtaining the maximum likelihood estimate of the parameters. Conversely, some of the methods available in **BayesLCA** are beyond the scope of frequentist approaches.

There are three main inferential functions in **BayesLCA**: an EM algorithm (Dempster, Laird, and Rubin 1977), a Gibbs sampler (Geman and Geman 1984) and a variational Bayes approximation (see Ormerod and Wand 2010, for an introduction to this method). Bishop (2006) contains excellent overviews of all three techniques. While each of the methods has been shown to be highly effective, associated difficulties can occur when attempting to perform inference. For example, label switching can often occur when using a Gibbs sampler, while headlong algorithms such as the EM algorithm and variational Bayes approximation can be sensitive to the starting values with which they are specified. While remedies for these difficulties are available, some may be more effective than others and the use of all three methods in combination may provide insight into the reliability of the statistical inferences being made. It is hoped that the paper may also serve as a useful introduction for those unfamiliar with the methods.

The rest of this paper is organized as follows: model specification for LCA is briefly outlined in Section 2. Demonstrations of the main inferential functions are provided in Sections 3, 4 and 5 respectively. These three sections each follow a similar format. Firstly, a broad overview of the inferential technique is given. The method is then applied to randomly generated synthetic data in order to illustrate its application in a simple manner. Some of the technical issues associated with the method are then discussed while being applied to the `Alzheimer` dataset included with the package. A description of both datasets is given in Section 2.1. Additional features are discussed in Section 6, and a summary of the package, as well as potential future extensions to the package are briefly discussed in Section 7.

All computations and graphics in this paper have been done with **BayesLCA** version 1.6. The latest version of the package should always be available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=BayesLCA`.

2. Latent class analysis

Latent class analysis involves performing inference within a mixture model framework, where the class of distribution is restricted to be a Bernoulli distribution. Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_N)$ denote M-dimensional vector-valued random variables, composed of G sub-populations, more commonly referred to as *classes* or *components*. Two sets of parameters, the G-dimensional vector $\boldsymbol{\tau}$ and $G \times M$ dimensional matrix $\boldsymbol{\theta}$, underly the model. These are referred to as the parameters for class and item probability respectively. The parameter τ_g denotes the prior probability of belonging to group g, so that $\tau_g \geq 0$ and $\sum_{g=1}^{G} \tau_g = 1$. The parameter θ_{gm} denotes the probability, conditional on membership of group g, that $X_{im} = 1$, for any $i \in 1, \ldots, N$, so that $p(X_{im}|\theta_{gm}) = \theta_{gm}^{X_{im}}(1 - \theta_{gm})^{1-X_{im}}$, for $X_{im} \in \{0, 1\}$. By making the *naïve Bayes* assumption (Hand and Yu 2001) that observations are conditionally independent based on group membership, the density of each $\mathbf{X}_i$ can then be written as

$$p(\mathbf{X}_i|\boldsymbol{\theta}, \boldsymbol{\tau}) = \sum_{g=1}^{G} \tau_g p(\mathbf{X}_i|\boldsymbol{\theta}_g), \tag{1}$$

where $p(\mathbf{X}_i|\boldsymbol{\theta}_g) = \prod_{m=1}^{M} p(X_{im}|\theta_{gm})$.

Direct inference using Equation 1 is typically difficult. The inference techniques in Sections 3, 4 and 5 are all predicated on the introduction of missing data $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_N)$. Each $\mathbf{Z}_i = (Z_{i1}, \ldots, Z_{iG})$ is a G-dimensional vector, representing the true class membership of $\mathbf{X}_i$ as a multinomial random variable. That is, suppose that the true group membership is known for each $\mathbf{X}_i$, and is denoted by

$$Z_{ig} := \begin{cases} 1 & \text{if observation } i \text{ belongs to group } g \\ 0 & \text{otherwise.} \end{cases}$$

The complete-data density for an observation $(\mathbf{X}_i, \mathbf{Z}_i)$ is then

$$p(\mathbf{X}_i, \mathbf{Z}_i|\boldsymbol{\tau}, \boldsymbol{\theta}) \quad = \quad \prod_{g=1}^{G} \left[\tau_g p(\mathbf{X}_i|\boldsymbol{\theta}_g)\right]^{Z_{ig}}.$$

Since $\mathbf{Z}_i$ is not known, the posterior probability for the class membership of observation i is given by

$$p(\mathbf{Z}_i|\mathbf{X}_i, \boldsymbol{\tau}, \boldsymbol{\theta}) \quad = \quad \prod_{g=1}^{G} \left[\frac{\tau_g p(\mathbf{X}_i|\boldsymbol{\theta}_g)}{\sum_{h=1}^{G} \tau_h p(\mathbf{X}_i|\boldsymbol{\theta}_h)}\right]^{Z_{ig}}.$$

To determine the complete-data posterior distribution $p(\boldsymbol{\tau}, \boldsymbol{\theta} | \mathbf{Z}_i, \mathbf{X}_i)$, we first assume conjugate prior distributions $p(\boldsymbol{\tau}|\boldsymbol{\delta})$ and $p(\theta_{gm}|\alpha_{gm}, \beta_{gm})$, for $\boldsymbol{\tau}$ and θ_{gm}, with corresponding hyperparameters $\alpha_{gm}, \beta_{gm}, \delta_g \in \mathbb{R}_+$ for each $g \in \{1, \ldots, G\}$ and $m \in \{1, \ldots, M\}$. That is, we assume that they follow Dirichlet and beta distributions respectively, which have the following forms:

$$\begin{aligned} p(\boldsymbol{\tau}|\boldsymbol{\delta}) &\propto \prod_{g=1}^{G} \tau_g^{\delta_g - 1} \\ p(\theta_{gm}|\alpha_{gm}, \beta_{gm}) &\propto \theta_{gm}^{\alpha_{gm}-1}(1-\theta_{gm})^{\beta_{gm}-1} \end{aligned}$$

for each $g \in \{1, \ldots, G\}$ and $m \in \{1, \ldots, M\}$.

Making these assumptions yields the posterior distribution (Garrett and Zeger 2000)

$$\begin{aligned} p(\boldsymbol{\tau}, \boldsymbol{\theta}|\mathbf{X}, \mathbf{Z}) &\propto \prod_{i=1}^{N} p(\mathbf{X}_i, \mathbf{Z}_i|\boldsymbol{\tau}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\boldsymbol{\tau}) \qquad (2) \\ &= \prod_{i=1}^{N}\prod_{g=1}^{G} \tau_g^{Z_{ig}+\delta_g-1} \prod_{m=1}^{M} \theta_{gm}^{X_{im}Z_{ig}+\alpha_{gm}-1}(1-\theta_{gm})^{(1-X_{im})Z_{ig}+\beta_{gm}-1}. \end{aligned}$$

Note that, as with all mixture models, latent class models are only identifiable up to a permutation of labels (Redner and Walker 1984; Leisch 2004). While this does not affect the inference methods discussed in Sections 3 and 5, issues do occur when employing Gibbs sampling methods, the details of which are discussed in Section 4.2.1. For convenience, once parameter estimates have been calculated in **BayesLCA**, labels are automatically indexed by decreasing order of size with respect to the class probability parameter $\boldsymbol{\tau}$, so that, e.g., group 3 will be the third largest group in the model.

2.1. Using BayesLCA

To begin, first load the **BayesLCA** package into R.

```
R> library("BayesLCA")
```

While not strictly necessary, setting the seed at the start will ensure identical results to those produced in this paper.

```
R> set.seed(123)
```

First, we simulate some data `X` to perform inference on. This can be done using the command `rlca()`. We generate a 500-row, 4-column matrix `X` with two underlying latent classes with parameters

$$\boldsymbol{\tau} = \begin{pmatrix} 0.4 & 0.6 \end{pmatrix} \text{ and } \boldsymbol{\theta} = \begin{pmatrix} 0.8 & 0.8 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.8 & 0.8 \end{pmatrix}$$

with the code

```
R> tau <- c(0.4, 0.6)
R> theta <- rbind(rep(c(0.8, 0.2), each = 2), rep(c(0.2, 0.8), each = 2))
R> X <- rlca(500, itemprob = theta, classprob = tau)
```

If `X` is inspected, it can be seen that many rows of the data are repeated. This stands to reason, as only $2^4 = 16$ unique combinations are possible. To store the data in a more efficient manner which is compatible with **BayesLCA**, use the command:

```
R> x <- data.blca(X)
```

This amounts to a list, consisting of a matrix of unique data patterns and a vector storing the number of times each pattern occurs.

We will also apply our methods to a dataset of patient symptoms recorded in the Mercer Institute of St. James' Hospital in Dublin, Ireland (Moran, Walsh, Lynch, Coen, Coakley, and Lawlor 2004; Walsh 2006). The data is a recording of the presence or absence of six symptoms displayed by 240 patients diagnosed with early onset Alzheimer's disease. Analysis of this dataset will highlight some of the technical issues associated with the inference techniques to be discussed. It is loaded into the R terminal using:

```
R> data("Alzheimer")
R> alz <- data.blca(Alzheimer)
```

3. EM algorithm

In an EM algorithm (Dempster *et al.* 1977), the expected complete-data log-posterior, which in this case is defined to be

$$Q(\boldsymbol{\theta}, \boldsymbol{\tau} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)}) := \mathsf{E}[\log p(\boldsymbol{\theta}, \boldsymbol{\tau} | \mathbf{X}, \mathbf{Z}) | \mathbf{X}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)}]$$

is iteratively maximised with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$, where $\boldsymbol{\theta}^{(k)}$ and $\boldsymbol{\tau}^{(k)}$ denote the values of $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ at iteration k (Tanner 1996). A lower bound of the observed-data log-posterior $\log p(\boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)} | \mathbf{X})$, each successive estimate of $\boldsymbol{\theta}^{(k)}$ and $\boldsymbol{\tau}^{(k)}$ with respect to $Q(\boldsymbol{\theta}, \boldsymbol{\tau} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)})$ in turn forces $\log p(\boldsymbol{\theta}, \boldsymbol{\tau} | \mathbf{X})$ to increase also, so that $\log p(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\tau}^{(k+1)} | \mathbf{X}) \geq \log p(\boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)} | \mathbf{X})$ whenever $Q(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\tau}^{(k+1)} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)}) \geq Q(\boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)})$. In this way local maxima for $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ are obtained (or a saddle point is reached). At the kth stage of the algorithm, parameter estimates are updated in two steps:

E-step: Compute $Q(\boldsymbol{\theta}, \boldsymbol{\tau} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)})$.

M-step: Set

$$\begin{aligned} \boldsymbol{\theta}^{(k+1)} &= \arg\max_{\boldsymbol{\theta} \in \Theta} Q(\boldsymbol{\theta}, \boldsymbol{\tau} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)}) \\ \boldsymbol{\tau}^{(k+1)} &= \arg\max_{\boldsymbol{\tau} \in T} Q(\boldsymbol{\theta}, \boldsymbol{\tau} | \boldsymbol{\theta}^{(k)}, \boldsymbol{\tau}^{(k)}). \end{aligned}$$

Here Θ and T denote the sample spaces for $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ respectively, i.e., Θ is the $G \times M$ dimensional unit hypercube, $\Theta = [0,1]^{G \times M}$, while T is the $G-1$ unit simplex

$$T = \left\{ \boldsymbol{\tau} = (\tau_1, \ldots, \tau_G) \in [0,1]^G \mid \sum_{g=1}^{G} \tau_g = 1 \right\}.$$

The E and M steps are repeated until the algorithm is deemed to converge.

In practice, the algorithm proceeds by executing:

E-step:

$$Z_{ig}^{(k+1)} = \frac{\tau_g^{(k)} p(\mathbf{X}_i|\boldsymbol{\theta}_g^{(k)})}{\sum_{h=1}^G \tau_h^{(k)} p(\mathbf{X}_i|\boldsymbol{\theta}_h^{(k)})}.$$

M-step:

$$\begin{aligned}
\theta_{gm}^{(k+1)} &= \frac{\sum_{i=1}^N X_{im} Z_{ig}^{(k+1)} + \alpha_{gm} - 1}{\sum_{i=1}^N Z_{ig}^{(k+1)} + \alpha_{gm} + \beta_{gm} - 2} \\
\tau_g^{(k+1)} &= \frac{\sum_{i=1}^N Z_{ig}^{(k+1)} + \delta_g - 1}{N + \sum_{h=1}^G \delta_h - G}.
\end{aligned}$$

Note that while estimation of $\boldsymbol{\tau}^{(k)}$ necessarily includes the constraint that the values sum to 1 (Bartholomew and Knott 1999), no such constraint is imposed to obtain $\boldsymbol{\theta}$. Only the binary nature of $\mathbf{X}$ and suitable prior values ensure that an estimate $\theta_{gm}^{(k)}$ lies in the interval $[0, 1]$. Posterior standard deviation estimates for the estimated parameter values may be obtained from the observed information (Bartholomew and Knott 1999; Linzer and Lewis 2011), or by using bootstrapping methods. These methods will be discussed further in Section 3.2. It is also possible to determine whether the algorithm has converged to a local maximum rather than a saddle point, by checking whether all eigenvalues of the observed information matrix are positive.

3.1. Synthetic data

Let's first use the EM algorithm method to estimate $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ from our synthetic data X, ignoring for the moment additional model output. The standard function to call when analysing data is `blca()`, and then using the argument `method` to specify how inference is to be performed, e.g., `method = "em"`. Alternatively, one can use `blca.` as a prefix, and then specify the method directly afterwards, for example `blca.em()`. To fit a 2-class model to the data using the EM method, with any additional arguments set to their default values, simply use the command:

```
R> fit1 <- blca(X, 2, method = "em")
R> fit1

MAP Estimates:

Item Probabilities:

        Col 1 Col 2 Col 3 Col 4
Group 1 0.244 0.226 0.755 0.760
Group 2 0.846 0.822 0.162 0.146

Membership Probabilities:
```

```
Group 1 Group 2
  0.651   0.349
Warning message:
Posterior standard deviations not returned.
```

Note that the item and membership probability estimates are close to the true values of $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$. We can also run the algorithm again, this time with prior values explicitly specified:

```
R> fit2 <- blca.em(x, 2, alpha = theta * 5, beta = (1 - theta) * 5,
+    delta = tau * 5)
```

Comparing the estimates for $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$ of `fit2` to `fit1`, we see that the results are highly similar, up to three decimal places. In this case, using a somewhat informative prior has had little impact on the result in comparison with a uniform, or non-informative, prior.

```
R> fit2$classprob

[1] 0.6507996 0.3492004

R> fit2$itemprob

          [,1]      [,2]      [,3]      [,4]
[1,] 0.2409866 0.2234701 0.7578669 0.7630877
[2,] 0.8492732 0.8253584 0.1584750 0.1431960
```

Use `names(fit1)` to see all items returned. The `blca.em` function returns an object with classes `"blca.em"` and `"blca"`, for which `print`, `summary` and `plot` methods exist. The `summary` command provides the prior values specified to the model, as well as other information, such as the number of iterations the algorithm ran for and the value of log-posterior found by the estimates.

```
R> summary(fit2)

Bayes-LCA
Diagnostic Summary
------------------

Hyper-Parameters:

 Item Probabilities:

 alpha:
        Col 1 Col 2 Col 3 Col 4
Group 1     1     1     4     4
Group 2     4     4     1     1

 beta:
        Col 1 Col 2 Col 3 Col 4
Group 1     4     4     1     1
Group 2     1     1     4     4

 Class Probabilities:

 delta:
Group 1 Group 2
      3       2
------------------

Method: EM algorithm

 Number of iterations: 32

 Log-Posterior Increase at Convergence: 0.0007128088

 Log-Posterior: -1237.986

 AIC: -2460.092

 BIC: -2498.024
```

Multiple figures may be produced by the `plot` method, some of which depend on the inference method used. These are specified by the argument `which`. Two figures common to all `blca` objects illustrate parameter values and classification uncertainty respectively. The first is a type of mosaic plot, where the y-axis denotes the size of each class, and the x-axis denotes each column of the data. Each cell's colour indicates the probability of a symptom being present, with high values having a bright yellow colour and low values having dark red. The second plot is another mosaic plot, with datapoints ordered along the x-axis according to size, and colour denoting class membership.

```
R> plot(fit1, which = 1:2)
```

These plots are shown in Figure 1. From inspection of the figure on the right, we can see that around half the points in the dataset are clustered with high levels of certainty, while the other half are still quite well distinguished.

3.2. `Alzheimer` data

We now perform an EM algorithm analysis to the `Alzheimer` data, confining our interest to 3-class models. While expert information may be available for the data, we shall assume uniform priors for all parameters.

Local maxima

A difficulty with EM algorithms is that while the log-posterior is increased at each iteration, they may converge to only a local maximum or saddle-point. To combat this, in `blca.em`

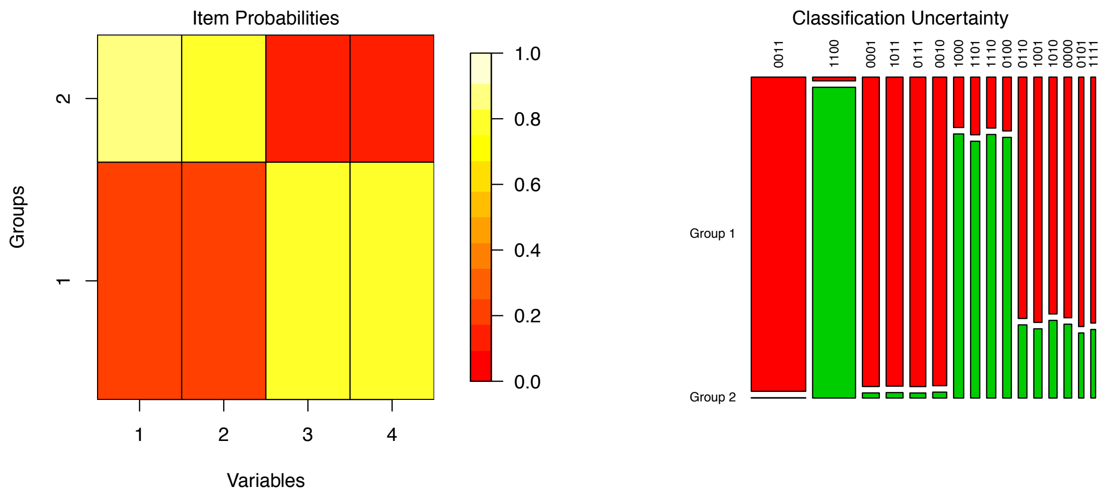

Figure 1: The figure on the left gives a visual indication of the underlying parameters of the model, while the figure on the right is a mosaic plot representing classification certainty.

the algorithm is restarted multiple times from randomly chosen starting values, keeping the set of parameters achieving the highest log-posterior value. See Section 6 for a description of the different ways which these values can be generated. The argument `restarts` is used to determine the number of times the algorithm should be run, with default setting `restarts = 5`. We perform an analysis of the data with a 3-class model. The following output is given by default. The warning message will be discussed in the next section:

```
R> sj3.em <- blca.em(alz, 3)

Restart number 1, logpost = -742.79...
Restart number 2, logpost = -744.28...
Restart number 3, logpost = -745.03...
Restart number 4, logpost = -745.26...
Restart number 5, logpost = -742.8...
Warning message:
In blca.em(alz, 3) :
  Some point estimates located at boundary (i.e., are 1 or 0).
  Posterior standard deviations will be 0 for these values.
```

From only five starts, the algorithm obtains three distinct local maxima of the log-posterior, each with a corresponding alternative set of parameter point estimates. If a sub-optimal set of estimates were incorrectly identified as obtaining the global maximum, then a very different interpretation of the dataset might be provided, potentially leading to a flawed analysis. In this case, it seems sensible to run the algorithm more times in order to identify the optimal parameters. The following code runs the algorithm for twenty times instead:

```
R> sj31.em <- blca.em(alz, 3, restarts = 20)
R> plot(sort(sj31.em$lpstarts), sequence(table(round(sj31.em$lpstarts, 2))),
+    main = "Converged Values", xlab = "Log-Posterior", ylab = "Frequency")
```

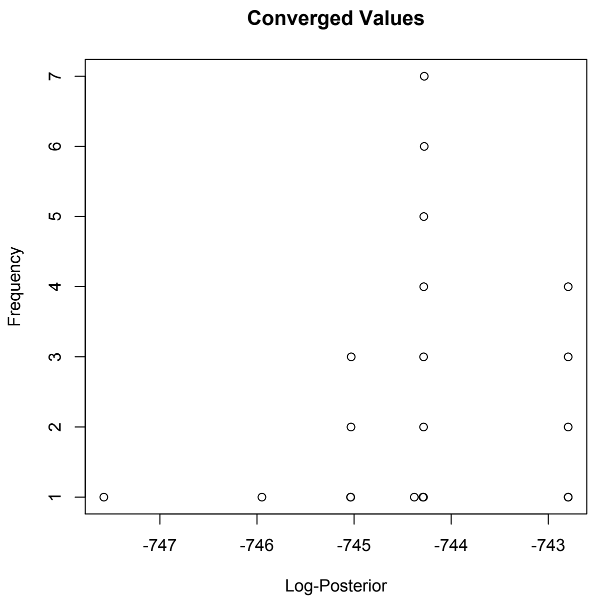

Figure 2: Point stacked histogram of values to which the EM algorithm runs converged. The algorithm appears to converge to several distinct sets of values.

A point stacked histogram of the log-posterior values is shown in Figure 2. This suggests that while the algorithm converges to several distinct sets of point estimates, the estimates attained by `sj31.em` do in fact appear to be the global maximum.

Posterior standard deviation estimation

So far, each model fitted using an EM algorithm has by default failed to return estimates of the posterior standard deviations. There are two ways of doing this in **BayesLCA**. Firstly, by using asymptotic methods. These can be obtained in two ways: setting the argument `sd = TRUE` when initially calling to `blca.em`, or, if the model has already been fit, by using the function `blca.em.sd`. The posterior standard deviation estimates for the model `fit1` fitted to the synthetic data `X` is given by the following code. Note that as well as the estimates, a convergence score is returned, in this case indicating that the algorithm converged to at least a local maximum, rather than a saddle point.

```
R> blca.em.sd(fit1, x)

$itemprob
           [,1]       [,2]       [,3]       [,4]
[1,] 0.02977234 0.02876512 0.02963475 0.03136158
[2,] 0.04178304 0.04354617 0.04197843 0.03859001

$classprob
[1] 0.03527256 0.03527144

$convergence
[1] 1
```

However, such a method is not without drawbacks. For example, point estimates occurring on the boundary must be obmitted from the observed information, as entries for estimates which are exactly 1 or 0 are undefined. Thus models such as `sj31.em` return posterior standard deviations of zero for some parameters when using this method. Unreliable posterior standard deviation estimates can also occur for parameter estimates occurring close to the boundary (Bartholomew and Knott 1999, Chapter 6).

```
R> blca.em.sd(sj31.em, alz)

$itemprob
           [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
[1,] 0.02869858 0.05846550 0.03293656 0.04448670 0.05696966 0.08370499
[2,] 0.03347575 0.05495001 0.06590206 0.08513284 0.05744349 0.00000000
[3,] 0.00000000 0.19756863 0.57852028 0.19389442 0.00000000 0.00000000

$classprob
[1] 0.09419236 0.08956211 0.01369711

$convergence
[1] 4
```

An alternative method of posterior standard deviation estimation employs bootstrapping methods (Wasserman 2007, Chapter 3). In particular, empirical bootstrapping methods involve sampling the data with replacement and re-fitting parameters to the new data. When done multiple times, posterior standard deviation of the parameter estimates may be obtained. It would also be possible to obtain estimates using parametric bootstrapping methods, whereby new datasets are generated from the fitted parameters of the model, and bootstrap samples then obtained by re-fitting to the newly generated data. However, this method may be comparatively unstable for our purposes; for example, when generating data from a model containing a group with low probability of membership, there is a non-negligible probability that some of the generated data samples will omit the group entirely.

For each bootstrapping run, values from the originally fitted model are used as the initial starting values for the EM algorithm, which is then run on the re-sampled data. This helps the algorithm to converge comparatively quickly over most samples. The use of these starting values should also help prevent any label-switching of parameter values from occurring during the bootstrapping run; a label-switching algorithm may also be used by specifying the argument `relabel = TRUE`, although this comes at an additional computational cost. See Section 4.2 for further details.

```
R> sj3.boot <- blca.boot(alz, fit =  sj31.em, B = 1000, relabel = TRUE)
R> sj3.boot

MAP Estimates:

Item Probabilities:
```

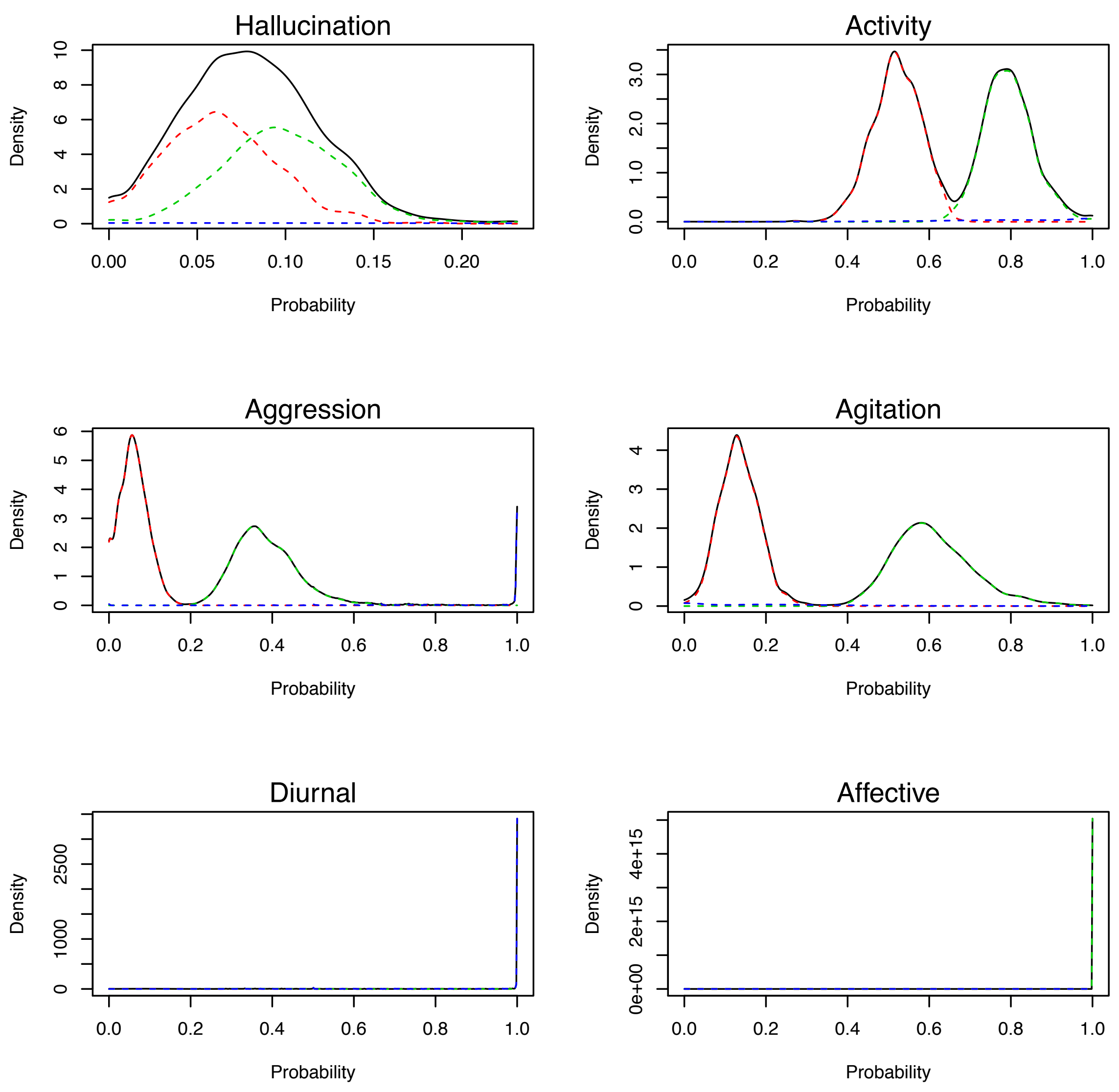

Figure 3: Density plots for the bootstrap estimates of the item probability parameters fitted by `sj3.boot`. Note the spikes in the plots for the Affective, Diurnal and Aggression symptoms. This indicates that the parameter estimates in question remained unchanged over all bootstrap samples.

```
        Hallucination Activity Aggression Agitation Diurnal Affective
Group 1         0.062    0.520      0.060     0.134   0.088     0.544
Group 2         0.098    0.793      0.384     0.610   0.372     1.000
Group 3         0.000    0.793      0.928     0.195   0.961     0.001

Membership Probabilities:

Group 1 Group 2 Group 3
  0.504   0.473   0.023
```

```
Posterior Standard Deviation Estimates:

Item Probabilities:

        Hallucination Activity Aggression Agitation Diurnal Affective
Group 1         0.032    0.059      0.035     0.047   0.047     0.090
Group 2         0.035    0.061      0.077     0.096   0.060     0.000
Group 3         0.000    0.225      0.166     0.209   0.127     0.029

Membership Probabilities:

Group 1 Group 2 Group 3
  0.089   0.088   0.011
```

Note that some Posterior standard deviations are exactly 0, indicating that identical parameter estimates were obtained from all bootstrap samples. `blca.boot` may be applied to a previously fitted object, or be called directly, with a new model fitted to the data as an initial step in the code. There is also a plot function, which includes density estimates for the parameters for item and class probability. Figure 3 shows the density estimates for the item probability parameters fitted by `sj3.boot`.

```
R> par(mfrow = c(3, 2))
R> plot(sj3.boot, which = 3)
```

4. Gibbs sampling

Gibbs sampling (Geman and Geman 1984) is a Markov chain Monte Carlo (MCMC) method typically used when direct sampling of the joint posterior distribution is intractable, but sampling from the full conditional distribution of each parameter is reasonably straightforward. By iteratively sampling from each conditional distribution in turn, samples of the joint posterior distribution are indirectly obtained. The method relies on the *Markov assumption*, in that samples drawn at iteration $k+1$ depend only on parameter values sampled during the previous iteration k:

- $\boldsymbol{\theta}^{(k+1)} \sim p(\boldsymbol{\theta}|\mathbf{X}, \boldsymbol{\tau}^{(k)}, \mathbf{Z}^{(k)}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$,
- $\boldsymbol{\tau}^{(k+1)} \sim p(\boldsymbol{\tau}|\mathbf{X}, \boldsymbol{\theta}^{(k+1)}, \mathbf{Z}^{(k)}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$,
- $\mathbf{Z}^{(k+1)} \sim p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{(k+1)}, \boldsymbol{\tau}^{(k+1)}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$.

In this way, parameter samples are repeatedly drawn, until it is decided that a reasonable representation of the joint posterior distribution has been obtained. The following draws are

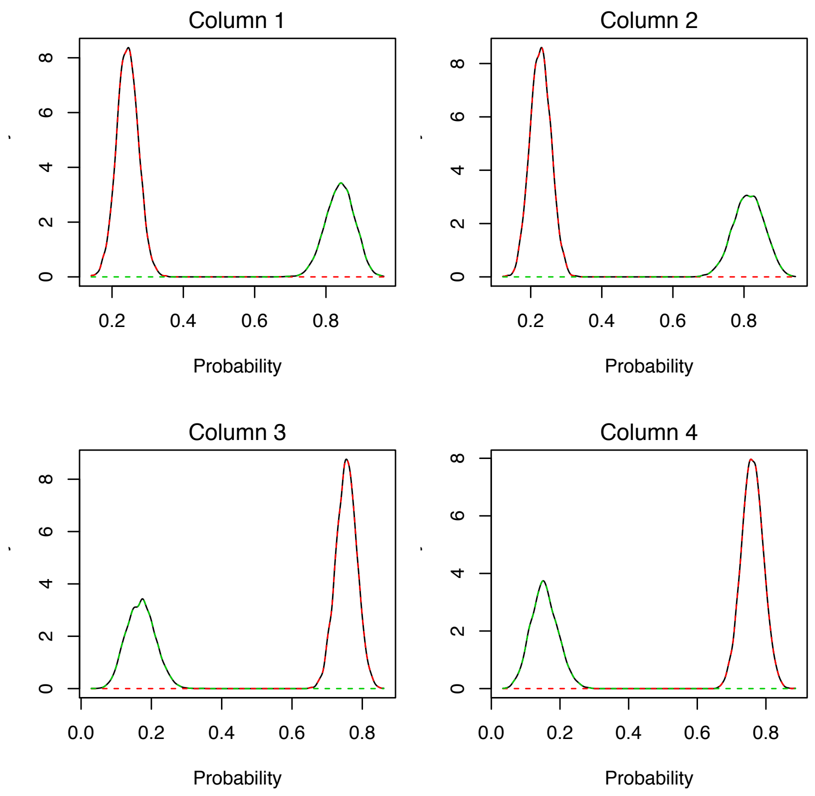

Figure 4: Density estimates of θ using Gibbs sampling.

thus made during a sample run (Garrett and Zeger 2000):

$$\begin{aligned}
\theta_{gm}^{(k+1)} &\sim \text{Beta}\left(\sum_{i=1}^{N} X_{im} Z_{ig}^{(k)} + \alpha_{gm}, \sum_{i=1}^{N} Z_{ig}^{(k)}(1 - X_{im}) + \beta_{gm}\right), \\
\boldsymbol{\tau}^{(k+1)} &\sim \text{Dirichlet}\left(\sum_{i=1}^{N} Z_{i1}^{(k)} + \delta_1, \ldots, \sum_{i=1}^{N} Z_{iG}^{(k)} + \delta_G\right), \\
\mathbf{Z}_i^{(k+1)} &\sim \text{Multinomial}\left(1, \frac{\tau_1^{(k+1)} p(\mathbf{X}_i|\boldsymbol{\theta}_1^{(k+1)})}{\sum_{h=1}^{G} \tau_h^{(k+1)} p(\mathbf{X}_i|\boldsymbol{\theta}_h^{(k+1)})}, \ldots, \frac{\tau_G^{(k+1)} p(\mathbf{X}_i|\boldsymbol{\theta}_G^{(k+1)})}{\sum_{h=1}^{G} \tau_h^{(k+1)} p(\mathbf{X}_i|\boldsymbol{\theta}_h^{(k+1)})}\right),
\end{aligned}$$

for each $g \in \{1, \ldots, G\}, m \in \{1, \ldots, M\}$ and $i \in \{1, \ldots, N\}$.

4.1. Synthetic data

Using a Gibbs sampler, parameters can be fit to the synthetic data using the following code:

```
R> fit2 <- blca.gibbs(x, 2)
R> plot(fit2, which = 3)
```

Density estimates for $\boldsymbol{\theta}$ are shown in Figure 4, with the true values of $\boldsymbol{\theta}$ clearly lying within areas of high density. Posterior standard deviation estimates are returned as standard.

4.2. `Alzheimer` data

Identifiability

To begin with, run the Gibbs sampler over its default settings, with a burn-in of 100 iterations and thinning rate of 1. The `relabel` setting will be explained shortly:

```
R> sj3.gibbs <- blca.gibbs(alz, 3, relabel = FALSE)
R> par(mfrow = c(4, 2))
R> plot(sj3.gibbs, which = 5)
```

A diagnostic plot of `sj3.gibbs` is shown in Figure 5. This provides clear evidence of label-switching, the phenomenon whereby group labels become permuted, causing multiple parameter spaces to be explored in the same run. There are many methods to deal with this problem: one approach (Stephens 2000; Celeux, Hurn, Robert, and P. 2000; Marin, Mengersen,

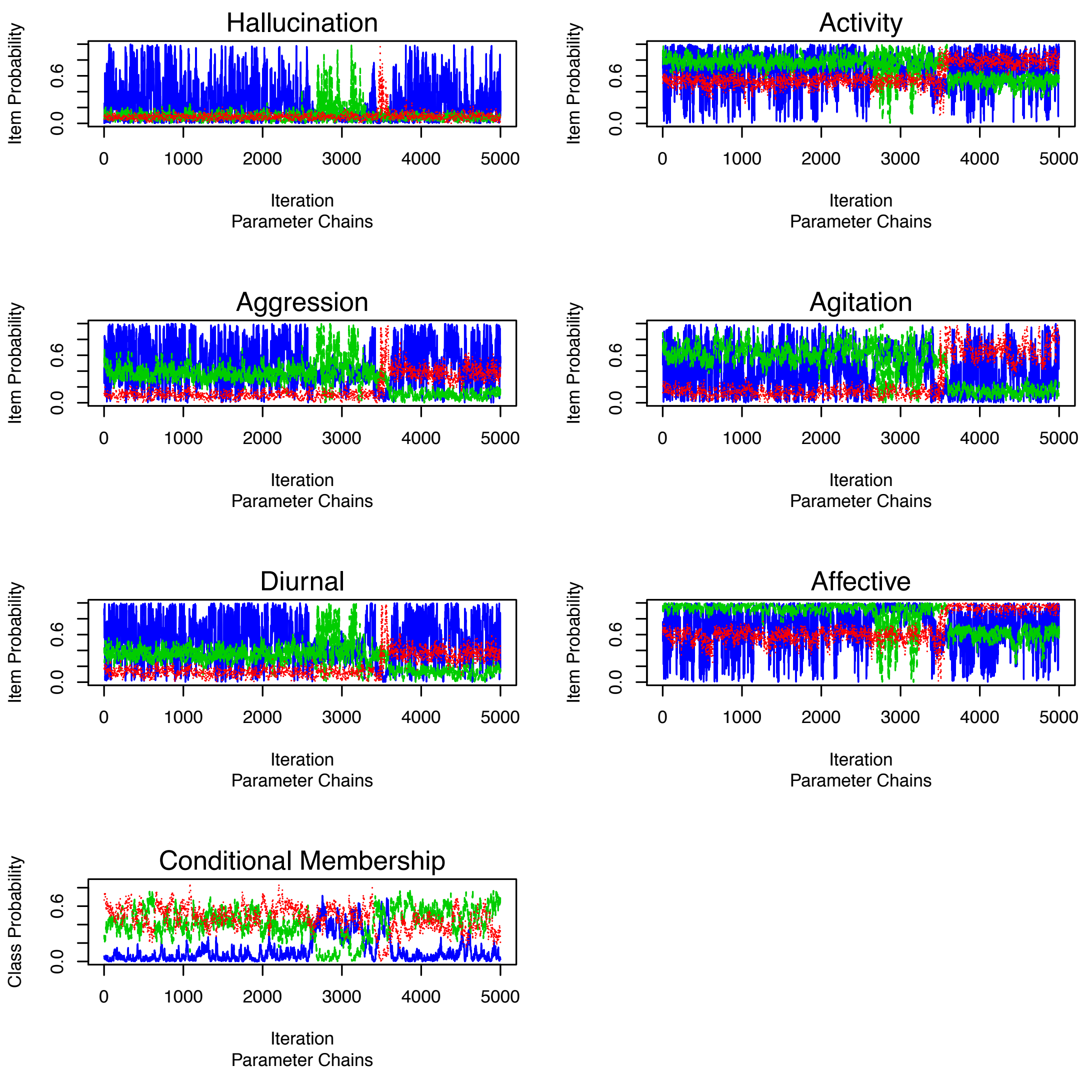

Figure 5: Clear evidence of label-switching taking place during the sample run.

and Robert 2005) is to employ a post-hoc re-labelling algorithm which attempts to minimise the posterior expectation of some loss function of the model parameters. Another approach is to impose an ordering constraint on parameters; however, this can lead to poor estimation (Celeux *et al.* 2000).

In **BayesLCA** a similar approach to Wyse and Friel (2012) is taken when dealing with this issue. The method re-labels samples by minimising a cost function of the group membership vector $\mathbf{Z}$. Let $\mathbf{Z}^{(1)}$ denote the first value of $\mathbf{Z}$ stored during the sample run. For any subsequent sample $\mathbf{Z}^{(j)}$, a $G \times G$ cost matrix $\mathbf{C}$ is created containing entries $C_{gh} = \sum_{n=1}^{N} Z_{ng}^{(1)} Z_{nh}^{(j)}$. The function `matchClasses()` from the **e1071** package (Dimitriadou *et al.* 2014) is then utilized to find the permutation σ so that the diagonal entries of $\mathbf{C}$ are maximised. In this way the agreement between $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(j)}$ is also maximised. The permuted values for $\boldsymbol{\theta}_\sigma$ and $\boldsymbol{\tau}_\sigma$ are then stored.

Burn-in, chain mixing

Two other key issues associated with Gibbs sampling are *burn-in* and *mixing*. Inspecting samples with respect to these terms helps to indicate the validity of the obtained model estimates. We again run the default model, except this time correcting for any relabelling which may occur.

```
R> sj30 <- blca.gibbs(alz, 3, relabel = TRUE)
```

Inspecting diagnostic plots of the model provides insight into how it has performed, without providing many clues as to how performance can be improved. One solution is to make use of diagnostic methods such as `raftery.diag`, available in the **coda** package (Plummer, Best, Cowles, and Vines 2006). This package is automatically loaded with **BayesLCA**, so there is no need to explicitly do so to make use of its functions. Objects of class `"blca.gibbs"` can be converted to type `"mcmc"` using the function `as.mcmc`.

```
R> raftery.diag(as.mcmc(sj30.gibbs))

Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95

              Burn-in  Total Lower bound  Dependence
              (M)      (N)   (Nmin)       factor (I)
 ClassProb 1  22       20778 3746          5.55
 ClassProb 2  30       33320 3746          8.89
 ClassProb 3  8        9488  3746          2.53
 ItemProb 1 1 12       12678 3746          3.38
 ItemProb 1 2 20       23284 3746          6.22
 ItemProb 1 3 10       11010 3746          2.94
 ItemProb 1 4 20       18042 3746          4.82
 ItemProb 1 5 20       27664 3746          7.38
 ItemProb 1 6 33       34095 3746          9.10
 ItemProb 2 1 12       12836 3746          3.43
```

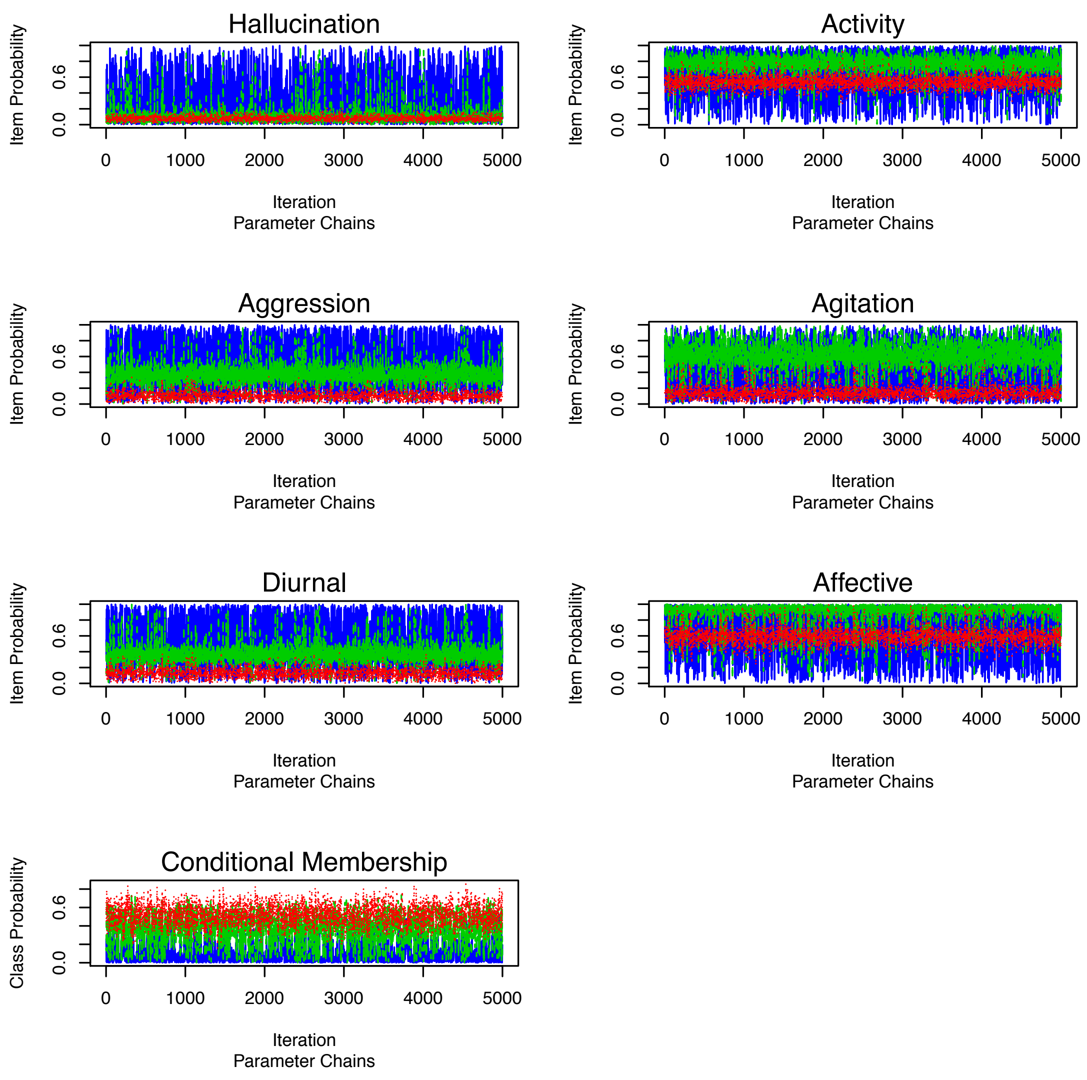

Figure 6: Diagnostic plot of the sampling run with label-switching corrected for, and better tuned parameters. These plots make clear the high uncertainty surrounding the item probability parameters of the third group (coloured blue).

```
ItemProb 2 2 8          10756 3746            2.87
ItemProb 2 3 50         48235 3746           12.90
ItemProb 2 4 24         29835 3746            7.96
ItemProb 2 5 18         20925 3746            5.59
ItemProb 2 6 12         14355 3746            3.83
ItemProb 3 1 4          4955  3746            1.32
ItemProb 3 2 6          6637  3746            1.77
ItemProb 3 3 15         16236 3746            4.33
ItemProb 3 4 4          5124  3746            1.37
ItemProb 3 5 9          11538 3746            3.08
ItemProb 3 6 5          5771  3746            1.54
```

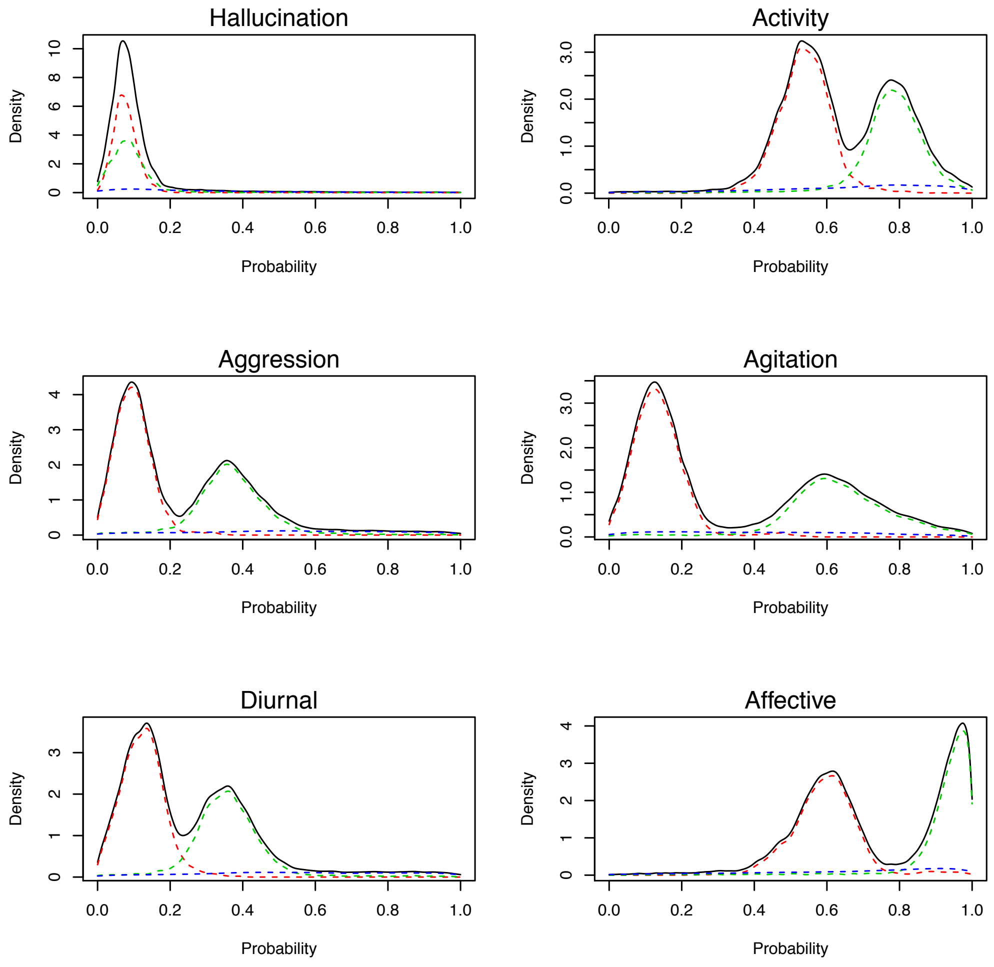

Figure 7: Posterior density plots for the item probability parameters with label-switching corrected for, and better tuned parameters. Groups are well separated for some symptoms, such as Aggression and Agitation, but not for others, notably Hallucination. Note the flatness of distributions from the third group (coloured blue), illustrating the lack of information gained by the model.

This output suggests that the sampler converges quickly (burn-in values are low), but is not mixing satisfactorily (note the high dependence factor of many parameters). A Gibbs sampler with better tuned parameters can then be run:

```
R> sj31.gibbs <- blca.gibbs(alz, 3, burn.in = 150, thin = 1/10, iter = 50000)
R> par(mfrow = c(3, 2))
R> plot(sj31.gibbs, which = 3)
```

Figure 6 shows a diagnostic plot for this model, created using the same code as for Figure 5. The behavior of the parameters in this case is much more satisfactory. Note that other functions in the **coda** package, such as the `plot` and `summary` methods for `"mcmc"` objects can also be used in this way. Density plots for $\boldsymbol{\theta}$ obtained from this sampling run are shown in Figure 7.

5. Variational Bayes

So far methods which attempt to maximize the joint posterior $p(\mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\tau} \mid \mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$, or iteratively sample from its conditional distributions have been discussed, in Sections 3 and 4 respectively. Variational Bayes methods can be thought of as an approximate combination of both techniques, in that they can be used to obtain parameter estimates which maximize a fully factorized posterior approximation to the joint posterior. In the general case, it can be shown that, for some latent parameters $\boldsymbol{\Omega}$ and arbitrary distribution q, a lower bound for the log-posterior $\log p(X)$ can always be obtained, via Jensen's inequality:

$$\begin{aligned}
\log p(X) &= \log \int p(\boldsymbol{\Omega}, \mathbf{X}) d\boldsymbol{\Omega} \\
&= \log \int p(\boldsymbol{\Omega}, \mathbf{X}) \frac{q(\boldsymbol{\Omega})}{q(\boldsymbol{\Omega})} d\boldsymbol{\Omega} \\
&\geq \int q(\boldsymbol{\Omega}) \log \frac{p(\boldsymbol{\Omega}, \mathbf{X})}{q(\boldsymbol{\Omega})} d\boldsymbol{\Omega} \\
&= \mathsf{E}_q \left[\log p(\boldsymbol{\Omega}, \mathbf{X})\right] - \mathsf{E}_q \left[\log q(\boldsymbol{\Omega})\right] \qquad (3)
\end{aligned}$$

with the discrepancy between the left and right hand sides of Equation 3 being equal to the Kullback-Leibler divergence $\mathrm{KL}(q||p)$ (Kullback and Leibler 1951). Supposing that $\boldsymbol{\Omega}$ can be partitioned into J parameters, such that $\boldsymbol{\Omega} = \{\Omega_1, \ldots, \Omega_J\}$, and restricting the family of distributions to which $q(\boldsymbol{\Omega})$ belongs to such that

$$q(\boldsymbol{\Omega}) = \prod_{j=1}^{J} q_j(\Omega_j),$$

it can be shown that the distribution q_j^* which minimises $\mathrm{KL}(q||p)$ has the form:

$$q_j^*(\Omega_j) \propto \exp\left\{\mathsf{E}_{i\neq j}\left[\log p(\mathbf{X}, \boldsymbol{\Omega})\right]\right\},$$

where the notation $\mathsf{E}_{i\neq j}\left[\log p(\mathbf{X}, \boldsymbol{\Omega})\right]$ denotes that the expectation of the log-posterior $\log p(\mathbf{X}, \boldsymbol{\Omega})$ is taken with respect to all model parameters excepting Ω_j, i.e., Ω_i, where $i = 1, \ldots, j-1, j+1, \ldots, J$.

In practice, the form of $q_j^*(\Omega_j)$ will be the same as that of the conditional distribution $p(\Omega_j \mid \mathbf{X}, \Omega_1, \ldots, \Omega_{j-1}, \Omega_{j+1}, \ldots, \Omega_J)$, with the key difference that its parameters are independent of the other variational parameters in the model. Updating each parameter iteratively, à la the EM algorithm, then maximizes $q(\boldsymbol{\Omega})$, and by extension $p(\mathbf{X}, \boldsymbol{\Omega})$.

The first step in applying this method to LCA is to introduce the variational distribution

$$q(\boldsymbol{\theta}, \boldsymbol{\tau}, \mathbf{Z}) = q(\boldsymbol{\theta} \mid \boldsymbol{\zeta}) q(\boldsymbol{\tau} \mid \boldsymbol{\gamma}) q(\mathbf{Z} \mid \boldsymbol{\phi}),$$

where $\boldsymbol{\zeta}, \boldsymbol{\tau}$ and $\boldsymbol{\phi}$ are all variational parameters. These have the following distributions, for each $i \in \{1, \ldots, N\}, g \in \{1, \ldots, G\}$, and $m \in \{1, \ldots, M\}$:

$$\begin{aligned}
\theta_{gm} \mid \zeta_{gm1}, \zeta_{gm2} &\sim \text{Beta}\left(\zeta_{gm1}, \zeta_{gm2}\right) \\
\boldsymbol{\tau} \mid \boldsymbol{\gamma} &\sim \text{Dirichlet}\left(\gamma_1, \ldots, \gamma_G\right) \\
\mathbf{Z}_i \mid \boldsymbol{\phi}_i &\sim \text{Multinomial}\left(\phi_{i1}, \ldots, \phi_{iG}\right),
\end{aligned}$$

and the variational parameters take the following updates:

$$
\begin{aligned}
\zeta_{gm1}^{(k+1)} &= \sum_{i=1}^{N} \phi_{ig}^{(k)} X_{im} + \alpha_{gm} \\
\zeta_{gm2}^{(k+1)} &= \sum_{i=1}^{N} \phi_{ig}^{(k)} (1 - X_{im}) + \beta_{gm} \\
\gamma_{g}^{(k+1)} &= \sum_{i=1}^{N} \phi_{ig}^{(k)} + \delta_g \\
\phi_{ig} &\propto \exp\left\{ \Psi\left(\gamma_g^{(k+1)}\right) - \Psi\left(\sum_{h=1}^{G} \gamma_h^{(k+1)}\right) + \sum_{m=1}^{M} X_{im}(\Psi(\zeta_{gm1}^{(k+1)}) - \Psi(\zeta_{gm1}^{(k+1)} + \zeta_{gm2}^{(k+1)})) \right. \\
&+ \left. \sum_{m=1}^{M} (1 - X_{im})(\Psi(\zeta_{gm2}^{(k+1)}) - \Psi(\zeta_{gm1}^{(k+1)} + \zeta_{gm2}^{(k+1)})) \right\},
\end{aligned}
$$

where Ψ denotes the *digamma* function (Abramowitz and Stegun 1965).

5.1. Synthetic and Alzheimer data

The following code fits 2- and 3-class models to the synthetic data `X` and `Alzheimer` data using variational Bayes methods:

```
R> fit3 <- blca.vb(x, 2)
R> sj3.vb <- blca.vb(alz, 3)
R> plot(fit3, which = 3)
```

Plots of density estimates for $\boldsymbol{\theta}$ are shown in Figure 8. Like in Section 4, the true values of $\boldsymbol{\theta}$ appear within areas of high posterior density, although the density estimates appear somewhat "pinched" in comparison with the plots in Figure 4. This can also be seen directly in the models fitted to the `Alzheimer` data:

```
R> sj3.vb$itemprob.sd

     Hallucination   Activity Aggression  Agitation    Diurnal  Affective
[1,]     0.02289787 0.04381683 0.02409615 0.03004852 0.02778058 0.04331132
[2,]     0.02873933 0.03830310 0.04598935 0.04591515 0.04576553 0.01537272
[3,]     0.14090801 0.16898290 0.15240892 0.16718027 0.14956479 0.15778048
```

```
R> sj31.gibbs$itemprob.sd

           [,1]      [,2]       [,3]       [,4]       [,5]       [,6]
[1,] 0.03223601 0.0726690 0.05115551 0.07451563 0.05871653 0.09384665
[2,] 0.09065832 0.1025723 0.11273175 0.15583202 0.11065335 0.11380141
[3,] 0.23396998 0.2357138 0.26530528 0.26388147 0.26755766 0.26072023
```

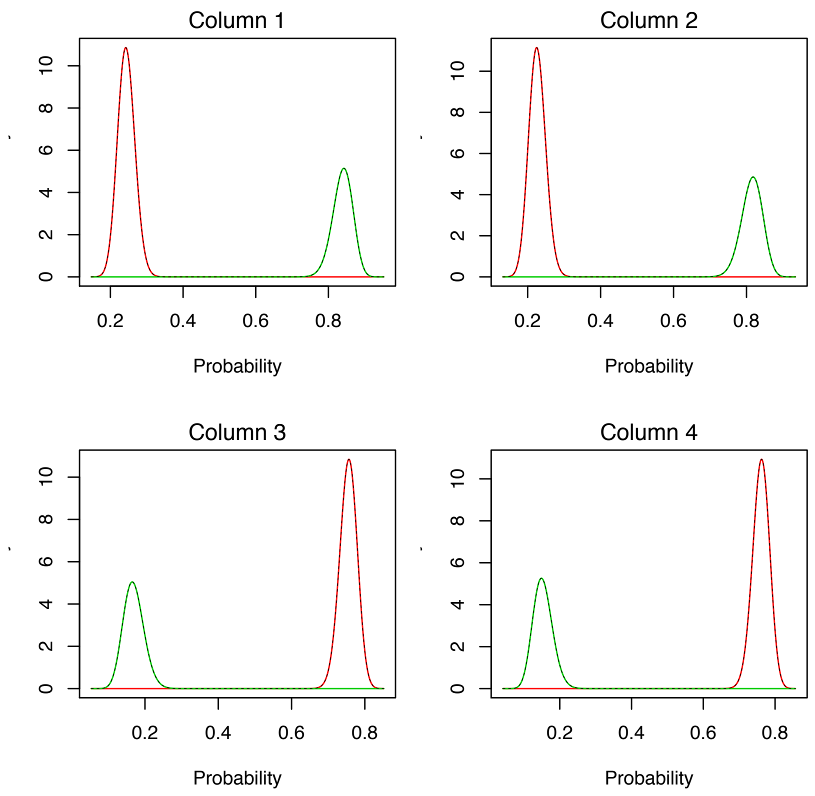

Figure 8: Approximate density estimates for θ using a variational Bayes approximation.

This is a common feature of variational Bayes approximations, in that the enforced independence between parameters results in diminished variance estimates. Conversely, it is this restriction which allows such quick density estimation.

Model selection

It is worth mentioning two additional properties of the variational Bayes approach which distinguish it from the EM algorithm. Firstly, that saddle point convergence issues , such as those discussed in Section 3.2, which are often encountered when fitting EM algorithms, are largely avoided (Bishop and Corduneanu 2001). The second is that, if the model is over-fitted, with Dirichlet prior values $\boldsymbol{\delta} < 1$ specified to the model, redundant components are emptied out, rather than the model becoming over-fitted (Bishop 2006). One method for determining an appropriate number of classes to fit to the `Alzheimer` data is to deliberately over-fit the model, and then consider only the classes for which $\tau_g > 0$ (Bishop and Corduneanu 2001). This is achieved by the following lines of code:

```
R> sj10.vb <- blca.vb(alz, 10, delta = 1/10)

Restart number 1, logpost = -1299.97...
Warning message:
In blca.vb(alz, 10, delta = 1/10) :
  Model may be improperly specified. Maximum number of classes that
  should be fitted is 9.
```

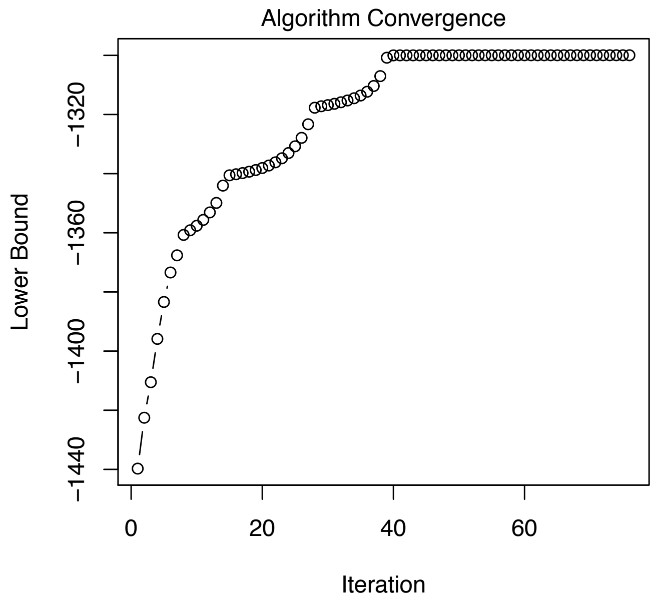

Figure 9: The sequence of lower bound values as the algorithm proceeds towards convergence.

```
R> sj10.vb$classprob
```

```
 [1] 0.5676355 0.4323645 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
 [8] 0.0000000 0.0000000 0.0000000
```

This suggests a 2-class fit is the best suited for the variational Bayes approximation.

The plotted values of the lower bound are shown in Figure 9. The multiple jumps in the lower bound indicate where components have "emptied" out.

```
R> plot(sj10.vb, which = 5)
```

6. Miscellaneous functions

6.1. Starting values

There are multiple ways to specify starting values in **BayesLCA**. This is done by specifying the `start.vals` option. The argument is used to assign initial values to the group membership vector $\mathbf{Z}$, either by specifying a method for how values are assigned, or by specifying the values directly. The default method is `start.vals = "single"`, whereby each unique data point is randomly assigned membership to a single class, that is,

$$\mathbf{Z}_i^{(0)} \sim \text{Multinomial}(1, 1/G, \ldots, 1/G), \text{ for each } i \in 1, \ldots, N.$$

Alternatively, specifying `start.vals= "across"` randomly assigns class membership across groups with respect to a uniform distribution. This is done in the following manner:

1. for $i \in 1, \dots, N$,

   for $g \in 1, \dots, G : W_{ig} \sim \text{Uniform}(0, 1)$.

2. Set $Z_{ig}^{(0)} = W_{ig} / \sum_{h=1}^{G} W_{ih}$.

As a quick comparison, consider the following case, where two 3-class models are fit to the `Alzheimer` data, using either type of starting value. The same random seed is specified, and the values of the log-posterior are then compared:

```
R> set.seed(123)
R> test1 <- blca.em(alz, 3, start.vals = "across", restarts = 1)
R> set.seed(123)
R> test2 <- blca.em(alz, 3, start.vals = "single", restarts = 1)
R> c(test1$logpost, test2$logpost)

[1] -745.0332 -744.2876
```

In this case, using single membership starting values provides a better fit.

Starting values may also be specified using the `Zscore` function. For example, when attempting to fit a 2-class model to the synthetic data `X`, class membership can be specified with respect to the true values of $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$:

```
R> Z1 <- Zscore(x$data, classprob = tau, itemprob = theta)
R> fit.true <- blca.em(x, 2, start.vals = Z1)
```

The returned output is almost identical to `fit1`, as fitted in Section 3.1.

Another practical use for specifying starting values would be to extend a Gibbs sampling run. By starting a run with values of $\mathbf{Z}$ sampled from the previous model, parameter samples may be drawn from the same region of sample space, negating the need for additional burn-in and, ideally, ensuring compatibility with the previous set of parameter samples.

```
R> Z2 <- apply(sj31.gibbs$Z, 1, rmultinom, size = 1, n = 1)
R> sj3new.gibbs <- blca.gibbs(alz, 3, iter = 10000, start.vals = t(Z2),
+    thin = 1/10, burn.in = 0)
```

6.2. Model selection

While the issue of over-fitting was discussed in the case of variational Bayes methods in Section 5.1, throughout Sections 3 and 4, the value of G, the number of underlying classes in the model, was assumed fixed and known. In this section, methods to determine the optimal number of classes to fit to a dataset using an EM algorithm or Gibbs sampling are discussed, with the `Alzheimer` data used as an illustrative example. Note that for latent class analysis, the number of classes which can be fit to data is at best limited by the condition $G(M+1) < 2^M$, or else the model becomes unidentifiable (Goodman 1974; Dean and Raftery 2010).

While the question of identifying the appropriate number of clusters to fit to a model remains an area of ongoing research, several methods involving information criteria have been developed. Such methods are predicated on testing:

$$\begin{aligned} H_0: &\qquad G = G_0 \\ H_1: &\qquad G = G_0 + 1 \end{aligned}$$

where G_0 is some positive integer (McLachlan and Peel 2002). For the EM algorithm, two such criteria are the Bayesian information criterion (BIC; Schwarz 1978; Kass and Raftery 1995) and Akaike's information criterion (AIC; Akaike 1973), while for Gibbs sampling, popular criteria include the deviance information criterion (DIC; Spiegelhalter, Best, Carlin, and van der Linde 2002), AIC Monte Carlo (AICM) and BIC Monte Carlo (BICM) (Raftery, Newton, Satagopan, and Krivitsky 2007). In all cases, the model with the higher value with respect to a criterion is considered the better fit to the data.

We will compare these criteria for a 1, 2, and 3-class model fit to the `Alzheimer` data. The interested reader is invited to apply these methods to the synthetic data `X` and investigate whether a 2-class model is selected.

EM algorithm

Firstly, we fit 1 and 2-class models to the data. In practice, fitting a 1-class model to the data amounts to separately fitting a Beta distribution to each column of the data in the standard manner, with the local independence assumption outlined in Section 2 replaced by the assumption that the data is independently, identically distributed. That is, $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{m=1}^{M} p(\theta_m \mid \mathbf{X}_m, \alpha_m, \beta_m)$, where each $p(\theta_m \mid \mathbf{X}_m, \alpha_m, \beta_m)$ follows a $\text{Beta}(\sum_{i=1}^{N} X_{im} + \alpha_m, \sum_{i=1}^{N}(1 - X_{im}) + \beta_m)$ distribution. For convenience, we fit the model the same way as the others :

```
R> sj1 <- blca.em(alz, 1, restarts = 1)
R> sj2.em <- blca.em(alz, 2)
```

Comparing the BIC and AIC of these models with that of the model `sj31.em` fitted in Section 3 gives the following output:

```
R> c(sj1$BIC, sj2.em$BIC, sj31.em$BIC)

[1] -1578.733 -1570.087 -1593.816

R> c(sj1$AIC, sj2.em$AIC, sj31.em$AIC)

[1] -1557.849 -1524.839 -1524.204
```

The BIC indicates that the 2-class model is selected as the optimal fit, while the difference between the AIC for the 2 and 3-class model is very small.

Gibbs sampling

In the case of Gibbs sampling, we first run a 2-class model using similarly tuned parameters to `sj31.gibbs` from Section 4. We then compare the DIC between 1-, 2- and 3-class models, using the fact that the DIC of the 1-class model is equal to twice its log-posterior:

```
R> sj2.gibbs <- blca.gibbs(alz, 2, thin = 1/10, iter = 50000, burn.in = 150)
R> c(2*sj1$logpost, sj2.gibbs$DIC, sj31.gibbs$DIC)

[1] -1545.849 -1522.593 -1517.245
```

This suggests that a 3-class fit may be best, although the difference between the 2- and 3-class models is small. Inspection of Figure 7 is also indicative of *weak identifiability* (Garrett and Zeger 2000), in that the posterior density of item probability parameters for the third group in the model are highly similar to their prior distributions (in this case, uniform Beta(1,1) distributions). This suggests that a 3-class model may be overfitting the data.

7. Discussion

In this paper we have demonstrated tools to perform LCA in a Bayesian setting using **BayesLCA**. The functions in this package do this by utilizing one of three methods: maximization of parameters *a posteriori*; sampling parameters via their conditional distributions; or by approximation of the joint posterior. While all three methods have been examined in detail, the computational cost of Gibbs sampling means that in many cases its implementation may be infeasible, particularly for data sets of a larger scale than those investigated here. Nevertheless, it remains something of a gold standard in terms of posterior estimation, and may be of benefit as a comparative tool, such as when the posterior standard deviation estimates of variational Bayes approximations were examined in Section 5. Future versions of the package may include a version of the method implemented using C code, which would substantially increase performance speed.

Currently, the package does not provide as many inference tools as **poLCA**: for example, it cannot be generalized to polytmous outcome variables, and cannot incorporate covariate information into a model. It is hoped to extend the package in the near future to include these features, so that a Bayesian alternative to such methods is available. A function to perform parametric bootstrapping may also be introduced, providing an alternative to the currently implemented non-parametric version.

In future versions of **BayesLCA** it may be of interest to include functions to perform collapsed Gibbs sampling (Nobile and Fearnside 2007). This has been successfully applied to latent block modeling (Wyse and Friel 2012), a class of models of which LCA is a subset. The method entails integrating out the item and class probability parameters $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$, and iteratively sampling from the posterior $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\delta})$. The method is primarily concerned with the clustering of data points, and parameter estimation can only be achieved via a post-hoc analysis. However, the comparative increase in speed between the collapsed and conventional sampler would be substantial. In addition the number of clusters can be included into the model as a random variable, providing an alternative method for model selection.

Acknowledgments

This work is supported by Science Foundation Ireland under the Clique Strategic Research Cluster (SFI/08/SRC/I1407) and the Insight Research Centre (SFI/12/RC/2289). The authors also wish to thank the article referees for providing many helpful comments and suggestions, both for this article and the **BayesLCA** package.

References

Abramowitz M, Stegun IA (1965). *Handbook of Mathematical Functions.* Dover Publications.

Akaike H (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In *Second International Symposium on Information Theory*, pp. 267–281. Akadémiai Kiadó.

Bartholomew DJ, Knott M (1999). *Latent Variable Models and Factor Analysis.* Kendall's Library of Statistics, 2nd edition. Hodder Arnold.

Benaglia T, Chauveau D, Hunter DR, Young D (2009). "**mixtools**: An R Package for Analyzing Finite Mixture Models." *Journal of Statistical Software*, **32**(6), 1–29. URL `http://www.jstatsoft.org/v32/i06/`.

Bishop CM (2006). *Pattern Recognition and Machine Learning.* Springer-Verlag, New York.

Bishop CM, Corduneanu A (2001). "Variational Bayesian Model Selection for Mixture Distributions." In T Jaakkola, T Richardson (eds.), *Proceedings Eighth International Conference on Artificial Intelligence and Statistics*, pp. 27–34. Morgan Kaufmann, Los Altos.

Celeux G, Hurn M, Robert, P C (2000). "Computational and Inferential Difficulties with Mixture Posterior Distributions." *Journal of the American Statistical Association*, **95**(451), 957–970.

Dean N, Raftery AE (2010). "Latent Class Analysis Variable Selection." *The Annals of the Institute of Statistical Mathematics*, **62**(1), 11–35.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

Dimitriadou E, Hornik K, Leisch F, Meyer D, Weingessel A (2014). ***e1071**: Misc Functions of the Department of Statistics (e1071), TU Wien.* R package version 1.6-2, URL `http://CRAN.R-project.org/package=e1071`.

Erosheva EA, Fienberg SE, Joutard C (2007). "Describing Disability through Individual-Level Mixture Models for Multivariate Binary Data." *The Annals of Applied Statistics*, **1**(2), 502–537.

Fraley C, Raftery A (2007). "Model-Based Methods of Classification: Using the **mclust** Software in Chemometrics." *Journal of Statistical Software*, **18**(6), 1–13. URL `http://www.jstatsoft.org/v18/i06/`.

Garrett ES, Zeger SL (2000). "Latent Class Model Diagnosis." *Biometrics*, **56**(4), 1055–1067.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(6), 721–741.

Goodman LA (1974). "Exploratory Latent Structure Analysis Using Both Identifiable and Unidentifiable Models." *Biometrika*, **61**(2), 215–231.

Hand DJ, Yu K (2001). "Idiot's Bayes: Not so Stupid after All?" *International Statistical Review*, **69**(3), 385–398.

Kass RE, Raftery AE (1995). "Bayes Factors." *Journal of the American Statistical Association*, **90**(430), 773–795.

Kullback S, Leibler RA (1951). "On Information and Sufficiency." *The Annals of Mathematical Statistics*, **22**(1), 79–86.

Leisch F (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8), 1–18. URL `http://www.jstatsoft.org/v11/i08/`.

Linzer DA, Lewis JB (2011). "**poLCA**: An R Package for Polytomous Variable Latent Class Analysis." *Journal of Statistical Software*, **42**(10), 1–29. URL `http://www.jstatsoft.org/v42/i10/`.

Marin JM, Mengersen K, Robert CP (2005). "Bayesian Modelling and Inference on Mixtures of Distributions." In D Dey, CR Rao (eds.), *Bayesian Thinking: Modeling and Computation*, volume 25 of *Handbook of Statistics*, chapter 16, pp. 459–507. North Holland, Amsterdam.

McLachlan G, Peel D (2002). *Finite Mixture Models*. John Wiley & Sons.

Moran M, Walsh C, Lynch A, Coen RF, Coakley D, Lawlor BA (2004). "Syndromes of Behavioural and Psychological Symptoms in Mild Alzheimer's Disease." *International Journal of Geriatric Psychiatry*, **19**(4), 359–364.

Nobile A, Fearnside A (2007). "Bayesian Finite Mixtures with an Unknown Number of Components: The Allocation Sampler." *Statistics and Computing*, **17**, 147–162.

Ormerod JT, Wand MP (2010). "Explaining Variational Approximations." *The American Statistician*, **64**(2), 140–153.

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/`.

Raftery AE, Newton MA, Satagopan JM, Krivitsky PN (2007). "Estimating the Integrated Likelihood via Posterior Simulation Using the Harmonic Mean Identity." In JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, M West (eds.), *Bayesian Statistics 8*, pp. 1–45. Oxford University Press, Oxford.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org/`.

Redner RA, Walker HF (1984). "Mixture Densities, Maximum Likelihood and the EM Algorithm." *SIAM Review*, **26**(2), 195–239.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464.

Spiegelhalter DJ, Best NG, Carlin BP, van der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society B*, **64**(4), 583–639.

Stephens M (2000). "Dealing with Label Switching in Mixture Models." *Journal of the Royal Statistical Society B*, **62**(4), 795–809.

Tanner MA (1996). *Tools for Statistical Inference.* Springer Series in Statistics, 3rd edition. Springer-Verlag, New York.

Walsh C (2006). "Latent Class Analysis Identification of Syndromes in Alzheimer's Disease: A Bayesian Approach." *Metodološki Zvezki – Advances in Methodology and Statistics*, **3**(1), 147 – 162.

Wasserman L (2007). *All of Nonparametric Statistics.* Springer Series in Statistics. Springer-Verlag.

Wyse J, Friel N (2012). "Block Clustering with Collapsed Latent Block Models." *Statistics and Computing*, **22**(1), 415–428.

Affiliation:

Arthur White
School of Mathematical Sciences
University College Dublin Belfield
Dublin 4, Ireland
E-mail: arthur.white@ucdconnect.ie
URL: https://sites.google.com/site/bayeslca/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 61, Issue 13 *Submitted:* 2012-06-13
November 2014 *Accepted:* 2014-07-11