bayesmix: An R package for Bayesian Mixture Modeling

Bettina Grün
Department of Statistics and Probability Theory
Vienna University of Technology

Abstract

The R package **bayesmix** provides the functionality for estimating univariate Gaussian mixture models with MCMC methods. Within a given model class users can modify the prior specifications and the initial values for developing a suitable model for their data. Furthermore, tools for analyzing the output of the MCMC simulations as, e.g., diagnostic plots, are available. The package is intended to be accompanying material for Frühwirth-Schnatter (2006). With this package the user can comfortably reproduce some of the results presented in the book. The data sets and functions for generating the initial values and prior specifications in the book are provided.

Keywords: finite mixture models, Bayesian modeling, MCMC methods, R.

1 Introduction

Finite mixture models are a popular method for modeling unobserved heterogeneity as well as for semi-parametric approximations of multimodal distributions. Areas of application are, e.g., biology, medicine, economics and engineering among many others. For maximum likelihood estimation the Expectation-Maximization (EM) algorithm is most frequently used which is provided in R for example by package **mclust** (Fraley and Raftery, 2002, 2006, revised 2009) or package **flexmix** (Grün and Leisch, 2008).

Bayesian estimation has become feasible with the advent of Markov chain Monte Carlo (MCMC) simulation and the R package **bayesmix** provides facilities for estimating univariate Gaussian finite mixtures with MCMC methods. The model class which can be estimated with **bayesmix** is a special case of a graphical model where the nodes and their distributions are fixed and the user only needs to specify the values of the constant nodes, the data and the initial values. Small variations of the model are allowed with respect to the segment specific priors. The MCMC sampling is done using package **rjags** (Plummer, 2009) which builds on JAGS (Just Another Gibbs Sampler; Plummer, 2003) and its output can be analyzed in R using functionality from the package **coda** (Plummer et al., 2009). In addition to visualization tools for the MCMC chains diagnostic plots are implemented which can be used to determine the appropriate number of segments or a suitable variable for ordering the segments. In Bayesian mixture modeling in general it makes a difference which constraint is imposed for ordering the segments due to label switching.

In Section 2 the model class which can be fitted with this package is described. In Section 3 a short overview on identifiability issues and MCMC estimation of this model class is given. In Section 4 we give a short description of the implementation and the most important functions. In Section 5 two different applications on well known data sets are presented. One application is on modeling heterogeneity in a fish population, while the other is on outlier modeling.

2 Model class

Finite Gaussian mixture distributions are given by

$$F(y_i) = \sum_{k=1}^{K} \eta_k \text{Normal}(y_i \,|\, \mu_k, \sigma_k^2),$$

where $\eta_k \geq 0$, $\sum_{k=1}^{K} \eta_k = 1$ and $\text{Normal}(\cdot \,|\, \mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2.

For Bayesian estimation of mixture models MCMC methods (Diebolt and Robert, 1994) are commonly used. The estimation process is often simplified by using data augmentation as pointed out by Dempster et al. (1977). For Bayesian estimation a prior distribution on the parameters has to be specified. The model class which is implemented in **bayesmix** only allows a certain structure of the prior distributions. In order to specify a model with another prior distribution the file containing the BUGS specification of the model has to be modified.

For the prior of η_k it is assumed that

$$\eta_1, \ldots, \eta_K \sim \text{Dirichlet}(e_{0,1}, \ldots, e_{0.K}).$$

For the priors on the parameters of the Gaussian distributions we distinguish between conditionally conjugate priors and independence priors, as well as the inclusion of hierarchical priors on the variances, where also partially proper priors can be specified.

For conditionally conjugate priors we have

$$\begin{aligned} \sigma_k^2 &\sim \text{InvGamma}(\frac{\nu_{0,k}}{2}, \frac{\nu_{0,k} S_{0,k}}{2}), \\ \mu_k \,|\, \sigma_k^2 &\sim \text{Normal}(b_{0,k}, B_{0,k}\sigma_k^2). \end{aligned}$$

For independence priors

$$\begin{aligned} \sigma_k^2 &\sim \text{InvGamma}(\frac{\nu_{0,k}}{2}, \frac{\nu_{0,k} S_{0,k}}{2}), \\ \mu_k &\sim \text{Normal}(b_{0,k}, B_{0,k}). \end{aligned}$$

For the hierarchical prior on the variances we add

$$S_{0,k} \sim \text{Gamma}(\frac{g_{0,k}}{2}, \frac{g_{0,k} G_{0,k}}{2}).$$

If improper priors are specified for μ_k or σ_k^2, the a-posteriori distributions might equally be improper because classes might be empty or include only elements of the same value. In Diebolt and Robert (1994) it was suggested to reject draws of S^N which lead to problems. However, this is complicated to achieve for **bayesmix** as JAGS samples each node separately and hence the classifications are not drawn together. Therefore, it is not recommended to use improper priors on μ_k or σ_k^2, which is not a severe drawback as the MCMC sampler has a much higher risk of being trapped at some local mode in such a setting (Frühwirth-Schnatter, 2006).

3 Model estimation

3.1 Identifiability

For mixture models there can be three different kinds of unidentifiability distinguished:

- label switching of the segments
- overfitting

- generic unidentifiability

The unidentifiability due to label switching can be eliminated by imposing a restriction on the ordering of the segments, as, e.g., the constraint that the means of the segment distributions are ascending. Overfitting leads to identifiability problems, as either the prior probabilities are not uniquely determined because of two equal segments or the segment distribution is not uniquely determined because of prior probability $\eta = 0$. Generic unidentifiability is not present for finite mixtures of Gaussian distributions (Teicher, 1963; Titterington et al., 1985). In order to avoid the problems caused by unidentifiability a constraint implying a unique ordering of the segments can be specified and the number of segments has to be properly chosen.

3.2 MCMC sampling

The following steps can be distinguished for the MCMC estimation of a mixture of Gaussian distributions:

1. Sample independently for each observation the group indicator S given the parameters of the segment distributions.

2. Sample the prior probabilities given the group indicators S.

3. Sample the parameters of the segment distributions given the data and the group indicators S.

The model class specified in Section 2 ensures that the conditional posterior distributions for each step can be handled within the conjugate setting.

4 Description of implementation

The package **bayesmix** uses package **rjags** to perform the MCMC sampling. **rjags** builds on JAGS, a stand-alone C++ program for Bayesian Graphical Modeling. JAGS stands for Just Another Gibbs Sampler and the motivation of its development was to clone BUGS (Bayesian inference Using Gibbs Sampling) in order to be able to modify the program if necessary as JAGS is open-source software. Furthermore, **bayesmix** uses functionality from package **coda**, which provides tools for output analysis and diagnostics of MCMC simulations. **bayesmix** depends on package **coda** because classes introduced in package **coda** are used for handling the MCMC chains, as e.g., to plot them.

bayesmix calls functions `jags.model()` and `coda.samples()` from package **rjags** for the MCMC sampling. The syntax which JAGS uses for specifying the models is similar to the one used in WinBUGS. A description of the WinBUGS language can be found in Spiegelhalter et al. (2003). The differences between WinBUGS and the syntax used by JAGS are described in the JAGS user manual (Plummer, 2004). The user manual also provides instructions for installing JAGS on Windows and Linux systems.

bayesmix allows the easy specification of a specific model within the given model class using `BMMmodel()` by selecting the appropriate prior structure and values and the initial values. When specifying the model the information is checked for consistency, as e.g., the dimension of the variables, and for inadmissible models. Note that it is for example not possible to estimate hierarchical prior models where the hyper-parameter follows an improper distribution without specifying an initial value for it. JAGS samples the nodes in a reverse partial ordering of the graph and therefore, the improper hyper-parameter is sampled after its children. A partial ordering of the nodes signifies that if a is after b in the ordering there is no path from a to b. A necessary condition for establishing a partial ordering is that the graph is acyclic.

With `JAGScontrol()` the number of burn-in and recorded draws, the seed and the recorded variables are determined. Given the model and the control `JAGScall()` creates the `.bug`-file and makes the MCMC simulations. The wrapper `JAGSrun()` also takes care of where the `.bug`-file is written to. By calling function `JAGScall()` it is possible to modify the `.bug`-file before MCMC sampling.

The results can be visually analyzed by using trace plots and by plotting the estimated kernel densities for the MCMC chains. Further diagnostic plots are provided which help to decide on the appropriate number of segments or on a reasonable constraint for ordering the segments. `BMMposteriori()` plots the estimated a-posteriori probabilities for the segments and can be used for determining how well the mixture defines disjoint clusters in the data.

Other functions allow the random permutation of the segments for each draw or the sorting of them with respect to a constraint specified for one of the variables. The random permutation of the draws allows to obtain similar results to those by a random permutation sampler, which has been proposed for inforcing balanced label switching in cases where the unconstrained likelihood shall be investigated (Frühwirth-Schnatter, 2001).

5 Application

In the following we demonstrate the use of **bayesmix** on two examples taken from Frühwirth-Schnatter (2006). The examples show different possible applications of mixture models. In the first example the capturing of unobserved heterogeneity is demonstrated, while in the second example outliers are detected.

5.1 Fishery data

We fit a Gaussian mixture model on the fishery data taken from Titterington et al. (1985). It consists of data on the lengths of 256 snappers. The heterogeneity in the data comes from the different age groups a fish might belong to depending if it comes from the current year's spawning or the previous, and so on.

We have to load the package and the data.

```
> library("bayesmix")
> data("fish", package = "bayesmix")
```

Then we have to specify the model: We choose $k = 4$ classes and assume an independent prior between μ and τ, where τ denotes $1/\sigma^2$. A hierarchical prior is specified for τ. The parameters for the prior for μ are $b_0 = \text{median(fish)}$ and $B_0^{-1} = 0$, whereas $\nu_0 = 20$ and S_0 follows a $\Gamma(0,0)$-distribution. Furthermore, we choose the default for the initial values which are equal to those suggested in Frühwirth-Schnatter (2006): equal values for η, corresponding quantiles for μ and equal values for τ which are determined by taking the inverse of a robust estimate of the variance (in our case IQR) divided by 1.34 squared.

Hence, the model can be specified by the function `BMMmodel()`:

```
> model <- BMMmodel(fish, k = 4, initialValues = list(S0 = 2),
+                   priors = list(kind = "independence",
+                     parameter = "priorsFish", hierarchical = "tau"))
```

The model specified in the BUGS syntax is given by:

```
> model

Data for nodes: b0, B0inv, nu0Half, g0Half, g0G0Half, k, N, e, y
Initial values for nodes: eta, mu, tau, S0

Model specification in BUGS language:

var
        b0,
         B0inv,
         nu0Half,
         g0Half,
         g0G0Half,
         k,
         N,
         eta[4],
        mu[4],
        tau[4],
        nu0S0Half,
         S0,
         e[4],
        y[256],
        S[256];

model        {
        for (i in 1:N) {
                y[i] ~ dnorm(mu[S[i]],tau[S[i]]);
                S[i] ~ dcat(eta[]);
        }
        for (j in 1:k) {
                mu[j] ~ dnorm(b0,B0inv);
                tau[j] ~ dgamma(nu0Half,nu0S0Half);
        }
        S0 ~ dgamma(g0Half,g0G0Half);
        nu0S0Half <- nu0Half * S0;

        eta[] ~ ddirch(e[]);
}
```

`JAGScontrol()` is used for specifying the control parameters for the MCMC sampling. We have to set the number of discarded burn-in draws, the number of monitored draws together with the names of the monitored variables and a possible seed of the RNG in order to be able to reproduce the results.

```
> control <- JAGScontrol(variables = c("mu", "tau", "eta", "S"),
+                        burn.in = 3000, n.iter = 2000, seed = 10)
```

In order to make the MCMC simulation `JAGSrun()` is called and returns an object of class `"JAGSrun"`.

```
> z <- JAGSrun(fish, model = model, control = control)
```

```
Compiling model graph
   Declaring variables
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 256
   Unobserved stochastic nodes: 266
   Total graph size: 1047

Initializing model
```

bayesmix provides diagnostic plots for `"JAGSrun"` objects in order to estimate the necessary number of segments and to choose a suitable variable for ordering the segments. Figure 1 indicates

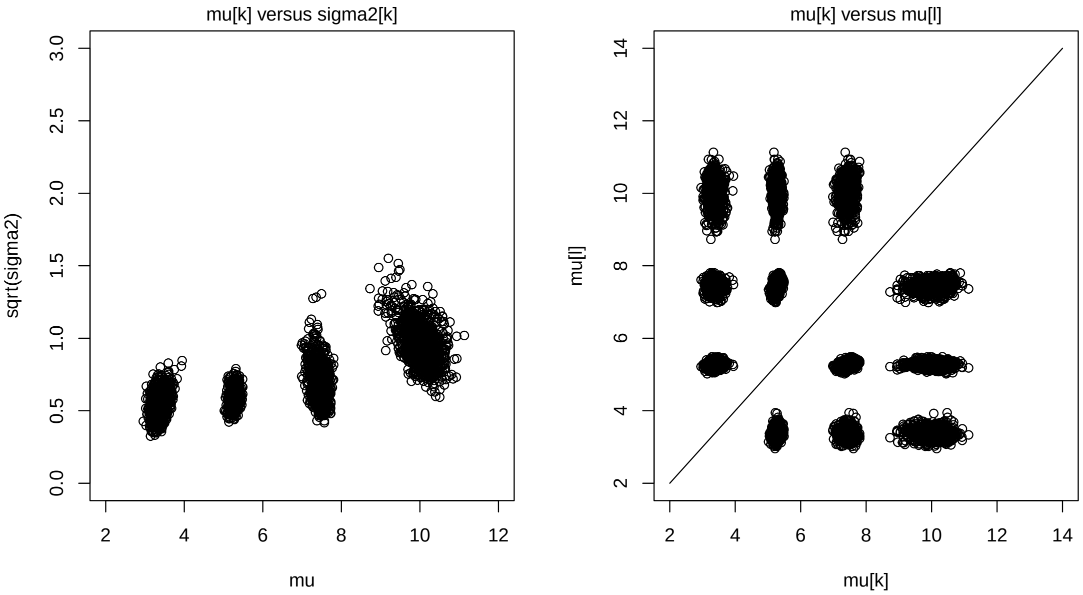

Figure 1: μ versus σ^2 and μ_k versus μ_l for $k \neq l$

that the segments are rather well separated for μ, as there are no points on the diagonal on the μ_k versus μ_l plot. From the μ versus σ plot it can be seen that σ is not a suitable variable for ordering the segments, while ordering of the segments with respect to μ is a sensible ordering constraint. Please note that only the last 1000 observations are plotted to reduce the document size.

Sorting the draws with respect to this constraint leads to the following results for the component specific parameters:

```
> zSort <- Sort(z, by = "mu")
> zSort

Call:
JAGSrun(y = fish, model = model, control = control)

Markov Chain Monte Carlo (MCMC) output:
Start = 3001
End = 5000
Thinning interval = 1

 Empirical mean, standard deviation and 95% CI for eta
          Mean      SD    2.5%  97.5%
eta[1] 0.11919 0.02621 0.07455 0.1783
eta[2] 0.50693 0.04422 0.41499 0.5911
eta[3] 0.27434 0.04396 0.18645 0.3647
eta[4] 0.09954 0.02974 0.05367 0.1698

 Empirical mean, standard deviation and 95% CI for mu
       Mean      SD  2.5%  97.5%
mu[1] 3.381 0.16156 3.125  3.726
mu[2] 5.277 0.07195 5.134  5.411
```

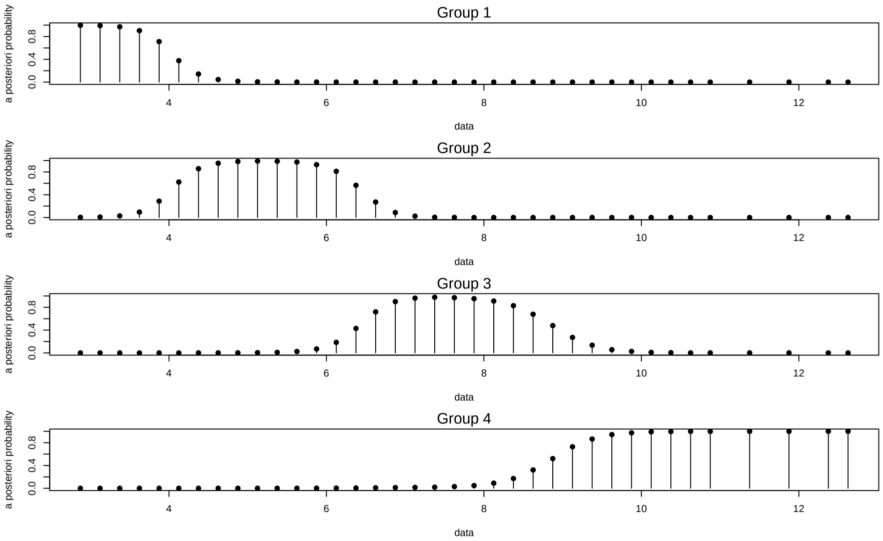

Figure 2: Posterior classification of the observations where the segments are sorted with respect to μ.

```
mu[3] 7.439 0.14116 7.127  7.691
mu[4] 9.977 0.40736 8.939 10.646

 Empirical mean, standard deviation and 95% CI for sigma2
           Mean      SD   2.5%  97.5%
sigma2[1] 0.2974 0.11378 0.1485 0.5816
sigma2[2] 0.3484 0.06468 0.2394 0.4971
sigma2[3] 0.4773 0.16578 0.2444 0.8712
sigma2[4] 0.9835 0.35779 0.5315 1.8928
```

After sorting the results the a-posteriori class probabilities of each observation can be estimated by the observed class affiliations over the runs, which can be seen in Figure 2. Furthermore, it is possible to make inference for the parameters of each segment. In Figure 3 there are, e.g., the MCMC traces and estimated densities for μ plotted.

Interesting modifications which can be made in the model specifications include the variation of the segment number k from 3 to 5 and different values for ν_0, e.g., 5 and 20. If ν_0 is chosen small the variance of the last segment is rather large which leads to high a-posteriori probabilities for the last segment of observations which lie between the modes of the precedent segments. Furthermore, the plot of μ_k versus μ_l indicates that μ does not provide a suitable identifiability constraint for $k = 4, 5$.

5.2 Darwin's data

Darwin's data consists of 15 observations of differences in heights between pairs of self-fertilized and cross-fertilized plants grown under the same condition. We demonstrate how finite mixture models can be used for robustifying Bayesian models with respect to outlying values.

For Darwin's data we fit a location-shift model in order to detect possible outliers. The location-shift model is a special case of a mixture of two normal distributions because it is a mixture of two normal distributions with different means but equal variances. For a location-shift model `restrict = "tau"` has to be specified for the model when calling `JAGSrun()`.

Furthermore, we want to ensure that there are no empty classes drawn which is specified in the model by `no.empty.classes = TRUE`.

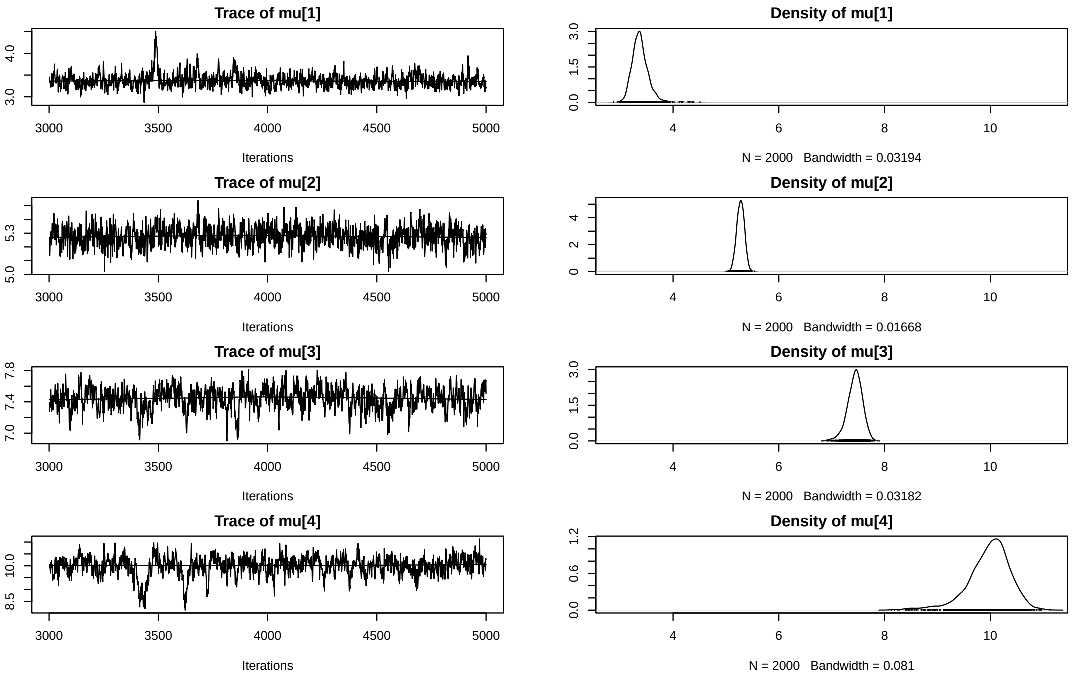

Figure 3: MCMC draws obtained for μ from the sampler under the constraint $\mu_1 < ... < \mu_4$.

We fit two different models, where in the first we assume equal a-priori class probabilities while in the second we assume that the second class is smaller. Under the assumption that with a probability of 95% at most 18.1% outliers are present in the data, a prior of $(15, 1)$ for the a-priori class probabilities is chosen.

```
> data("darwin", package = "bayesmix")
> k <- 2
> variables <- c("mu", "tau", "eta", "S")
> zBeta1.1 <- JAGSrun(darwin,
+                     model = BMMmodel(k = k,
+                       priors = list(kind = "independence"),
+                       no.empty.classes = TRUE, restrict = "tau"),
+                     control = JAGScontrol(variables, n.iter = 3000,
+                       burn.in = 1000, seed = 1))

Compiling model graph
   Declaring variables
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 17
   Unobserved stochastic nodes: 19
   Total graph size: 86

Initializing model

> zBeta1.15 <- JAGSrun(darwin,
+                      model = BMMmodel(k = k,
+                        priors = list(kind = "independence"),
```

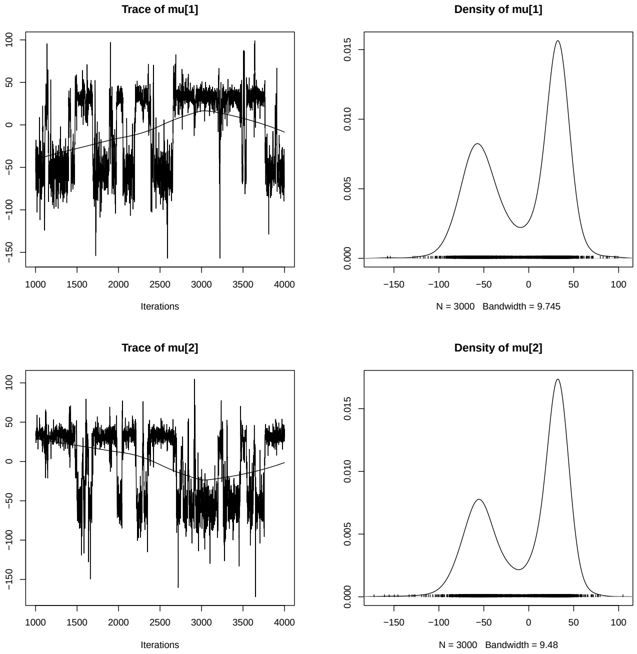

Figure 4: MCMC traces and posterior densities for μ for the location-shift model with Dirichlet$(1, 1)$-prior

```
+                          aprioriWeights = c(15, 1), no.empty.classes = TRUE,
+                          restrict = "tau"),
+                        control = JAGScontrol(variables, n.iter = 3000,
+                          burn.in = 1000, seed = 1))

Compiling model graph
   Declaring variables
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 17
   Unobserved stochastic nodes: 19
   Total graph size: 86

Initializing model
```

If we compare the results for the symmetric prior with those for the Dirichlet$(15, 1)$-prior we can see that label switching occurs frequently for μ for the symmetric prior whereas this is not the case for the asymmetric one (cp. Figure 4 and 5). The label switching for the symmetric prior leads to classification probabilities extremely biased towards 0.5 (cp. Figure 7).

6 Conclusions

bayesmix allows a user who "only" wants to estimate finite mixtures of Gaussian distributions to use **rjags** and hence JAGS as sampling engine without knowing the BUGS syntax which is used by JAGS for specifying general Bayesian hierarchical models. **bayesmix** offers the opportunity to be

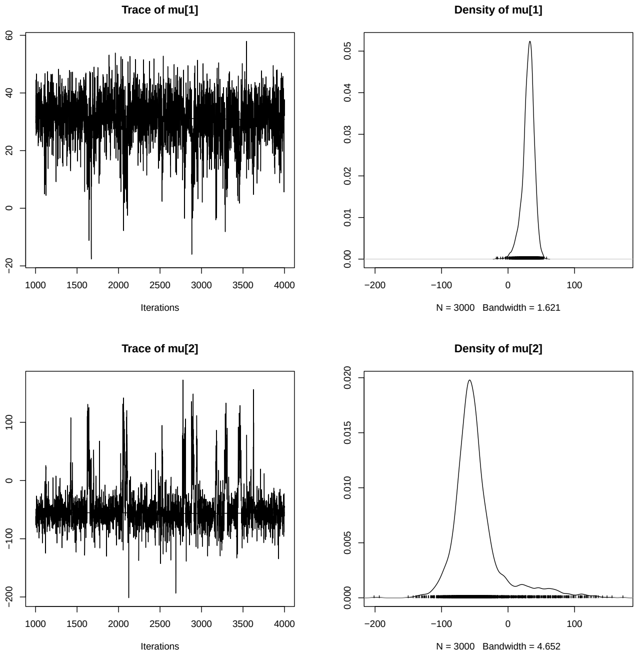

Figure 5: MCMC traces and posterior densities for μ for the location-shift model with Dirichlet$(15, 1)$-prior

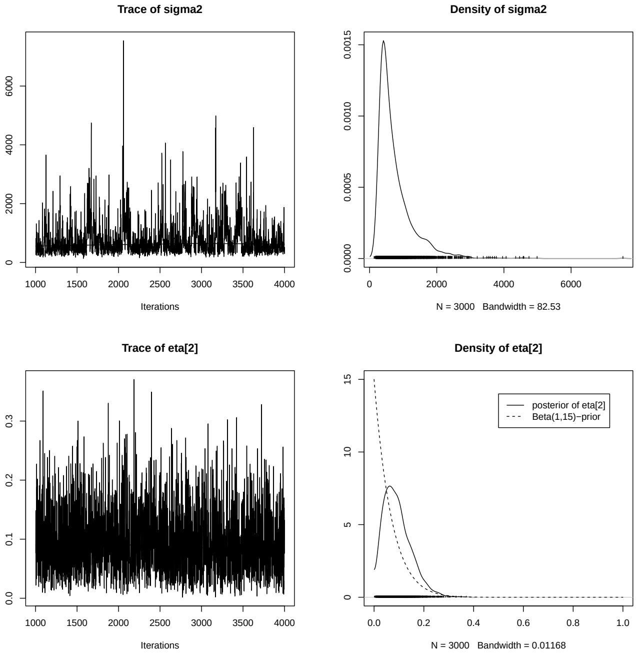

Figure 6: MCMC traces and posterior densities for σ^2 and η for the location-shift model with Dirichlet$(15, 1)$-prior

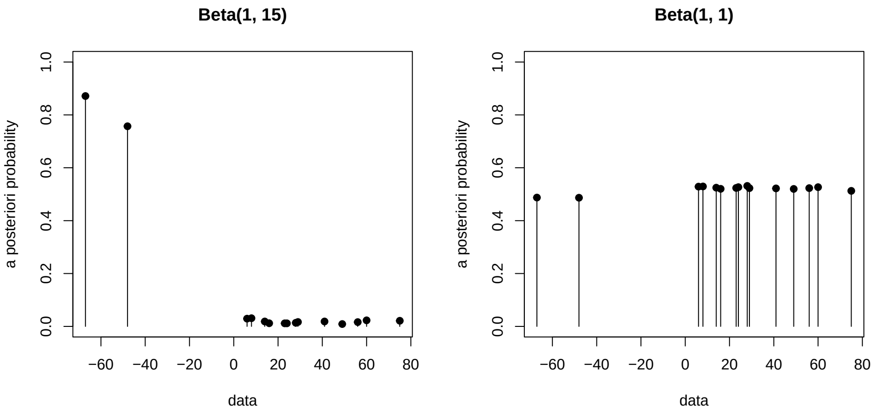

Figure 7: Outlier classification for Darwin's data

a starting point for learning the BUGS syntax as the model specification is written into a separate file and can be inspected and modified by the user.

References

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM-alogrithm. *Journal of the Royal Statistical Society, B*, 39:1–38, 1977.

J. Diebolt and C. P. Robert. Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society, Series B*, 56:363–375, 1994.

Chris Fraley and Adrian E. Raftery. Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2002.

Chris Fraley and Adrian E. Raftery. MCLUST version 3 for R: Normal mixture modeling and model-based clustering. Technical Report 504, University of Washington, Department of Statistics, 2006, revised 2009.

Sylvia Frühwirth-Schnatter. Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. *Journal of the American Statistical Association*, 96(453):194–209, 2001.

Sylvia Frühwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer Series in Statistics. Springer, New York, 2006. ISBN 0-387-32909-9.

Bettina Grün and Friedrich Leisch. FlexMix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, 28(4):1–35, 2008. URL `http://www.jstatsoft.org/v28/i04/`.

Martyn Plummer. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In Kurt Hornik, Friedrich Leisch, and Achim Zeileis, editors, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, Technische Universität Wien, Vienna, Austria, 2003. URL `http://www.ci.tuwien.ac.at/Conferences/DSC.html`. ISSN 1609-395X.

Martyn Plummer. *JAGS Version 0.50 Manual*. International Agency for Research on Cancer, April 2004. URL `http://www-fis.iarc.fr/~martyn/software/jags/`.

Martyn Plummer. *rjags: Bayesian Graphical Models Using MCMC*, 2009. URL http://CRAN.R-project.org/package=rjags. R package version 1.0.3-13.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. *coda: Output Analysis and Diagnostics for MCMC*, 2009. R package version 0.13-4.

David. J. Spiegelhalter, Andrew Thomas, Nicky G. Best, and Dave Lunn. *WinBUGS User Manual, Version 1.4*, 2003. URL http://www.mrc-bsu.cam.ac.uk/bugs/.

H. Teicher. Identifiability of finite mixtures. *Annals of Mathematical Statistics*, 34:1265–1269, 1963.

D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Chichester: Wiley, 1985.