Bayesian analysis of finite mixture distributions using the allocation sampler

Alastair T Fearnside

A Dissertation Submitted to the
Faculty of Information and Mathematical Sciences
at the University of Glasgow
for the degree of
Doctor of Philosophy

Department of Statistics

July 2007

Abstract

Finite mixture distributions are receiving more and more attention from statisticians in many different fields of research because they are a very flexible class of models. They are typically used for density estimation or to model population heterogeneity. One can think of a finite mixture distribution as grouping the observations into components from which they are assumed to have arisen. In certain settings these groups have a physical interpretation. The interest in these distributions has been boosted recently because of the ever increasing computer power available to researchers to carry out the computationally intensive tasks required in their analysis.

In order to fit a finite mixture distribution taking a Bayesian approach a posterior distribution has to be evaluated. When the number of components in the model is assumed known this posterior distribution can be sampled from using methods such as Data Augmentation or Gibbs sampling (Tanner and Wong (1987) and Gelfand and Smith (1990)) and the Metropolis-Hastings algorithm (Hastings (1970)). However, the number of components in the model can also be considered an unknown and an object of inference. Richardson and Green (1997) and Stephens (2000a) both describe Bayesian methods to sample across models with different numbers of components. This enables an estimate of the posterior

distribution of the number of components to be evaluated. Richardson and Green (1997) define a reversible jump Markov chain Monte Carlo (RJMCMC) sampler while Stephens (2000a) uses a Markov birth-death process approach sample from the posterior distribution. In this thesis a Markov chain Monte Carlo method, named the allocation sampler. This sampler differs from the RJMCMC method reported in Richardson and Green (1997) because the state space of the sampler is simplified by the assumption that the components' parameters and weights can be analytically integrated out of the model. This in turn has the advantage that only minimal changes are required to the sampler for mixtures of components from other parametric families. This thesis illustrates the allocation sampler's performance on both simulated and real data sets.

Chapter 1 provides a background to finite mixture distributions and gives an overview of some inferential techniques that have already been used to analyse these distributions.

Chapter 2 sets out the Bayesian model framework that is used throughout this thesis and defines all the required distributional results.

Chapter 3 describes the allocation sampler.

Chapter 4 tests the performance of the allocation sampler using simulated datasets from a collection of 15 different known mixture distributions.

Chapter 5 illustrates the allocation sampler with real datasets from a number of different research fields.

Chapter 6 summarises the research in the thesis and provides areas of possible future research.

Acknowledgements

I would like to take this opportunity to thank everyone who has helped me complete this thesis. Firstly, to my supervisor, Dr Agostino Nobile, who contributed his time and expertise to this thesis. I must also give thanks to him for passing on his wealth of knowledge and giving me continued support and encouragement when completing this work. I would also like to thank my second supervisor Prof Mike Titterington for the helpful comments he has given on this work. My Mum and her red pen also get a big thank you! Thanks must also go to all the other members of the Statistics department who have helped make the last few years a very enjoyable experience. I am very grateful for all the opportunities the department gave me. Also, I must thank the Engineering and Physical Sciences Research Council for funding me throughout this research.

I would also like to take this chance to thank the University of Glasgow Golf Club for giving me an activity to take my mind off the sometimes stressful work. I'm very grateful for the many opportunities to totally relax on some of the most famous links in the world.

Finally, I would like to thank my family and friends for all that they have done for me over these years. However, I must pay a special thanks to my Mum and Dad for all their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Analysis of finite mixture models can be dated back to the late 19th century when the Pearson (1894) paper was published. It contained an analysis of a mixture of two normal components on the well known crab data set. A finite mixture density can be described as a convex combination of a finite number of probability densities $q(x|\theta)$

$$f(x) = \sum_{j=1}^{k} \lambda_j q_j(x|\theta_j), \tag{1.1}$$

where the weights $\lambda_j > 0$ and $\sum_{j=1}^{k} \lambda_j = 1$. The term mixture component is used for $q_j(x|\theta_j)$, where this is a density that comes from a parametric family. The other parameters that are in model (1.1) are the number of components k, the mixture weights $\lambda = (\lambda_1, \ldots, \lambda_k)$ and the parameters corresponding to each of the components $\theta = (\theta_1, \ldots, \theta_k)$.

Due to their flexibility, mixtures can be used to model complex probability distributions that are not easily described using standard models. In particular,

they can be used to model population heterogeneity. Furthermore, they can be applied in many different ways, including density estimation, latent class analysis and cluster analysis. Therefore, they are being used to model data arising in many different fields of research, as diverse as astronomy, (e.g. velocities of galaxies), to philately (e.g. the thickness of Mexican stamps). However, they were neglected for a long while by statisticians because of the computationally expensive tasks required in their analysis. For example, Pearson had to solve a nonic equation to analyse just a mixture of two normal distributions. Mixture models have however seen a real boost in popularity in recent years, due to the tremendous increase in available computer power. This computer power is required to apply many of the Markov chain Monte Carlo (MCMC) methods that are becoming available at an ever increasing rate to fit the model using a Bayesian approach. There have been a number of books and monographs published over the last thirty years detailing the capabilities and analysis of mixture models, including Everitt and Hand (1981), Titterington et al. (1985), MacLachlan and Basford (1987) and MacLachlan and Peel (2000). A recent publication that summarises the most popular methods available to analyse these models is Marin et al. (2005).

In Bayesian inference, Bayes' theorem is used to convert prior beliefs about a parameter θ, to posterior beliefs, when we observe some data $x = (x_1, \ldots, x_n)$:

$$f(\theta|x) = \frac{f(\theta)L(\theta; x)}{\int f(\theta)L(\theta; x)d\theta} \tag{1.2}$$

where $f(\theta)$ is called the prior distribution, $L(\theta; x)$ is the likelihood function and $f(\theta|x)$ is the posterior distribution. Note that the role of the denominator is to make the posterior distribution integrate to 1. The prior distribution is the form

in which prior knowledge about the parameter of interest, θ, is incorporated into the model. If there is no prior knowledge, then a non-informative prior may be used, and thus the posterior will be strongly linked to the likelihood.

The Bayesian approach for parameter estimation hasn't always had so much interest. The first method used to estimate the parameters of a model of the type defined in (1.1) was the method of moments. This was the method used in Pearson (1894) and was the estimation tool of choice until Rao (1948) suggested the use of maximum likelihood estimation. Then, to enhance the backing for the use of maximum likelihood methods, the papers by Tan and Chang (1972) and Fryer and Robertson (1972) both showed evidence that they lead to more accurate results than those using the method of moments when estimating the parameters for most mixtures of normals. A major breakthrough in the maximum likelihood approach came with the publication of Dempster et al. (1977). This paper defines a general method for computing maximum likelihood estimates for missing data problems. They also define a finite mixture model as a missing data problem, where the missing data are unobserved vectors that indicate from which component each observation has arisen. This algorithm, known as the EM algorithm, has very widespread applicability. It is an iterative procedure that consists of two steps, the Expectation step and Maximisation step. It works in the finite mixture framework by calculating the expectation of the complete-data log-likelihood, conditional on the observed data and the current values for the parameters. Then, this expectation is maximised with respect to the parameters to give an improved set of parameters that are then used in the next iteration. An important property of this algorithm is that, at every stage, the log-likelihood corresponding to the observed data is improved. However, there are drawbacks

in that it converges slowly to a maximum, and also the maximum to which it converges is not necessarily the global maximum. Therefore, there have been attempts to advance the algorithm and variations are defined in numerous papers. Meng and Rubin (1993), Liu and Rubin (1994), Meng and van Dyk (1997) and Neal and Hinton (1998) define some variations to the standard EM algorithm that increase the speed of convergence and try to reduce the complexity of the sometimes complicated maximisation step. A comprehensive summary of the EM algorithm and its extensions is given by MacLachlan and Krishnan (1997). A final point to note is that special cases of this algorithm had been implemented well before Dempster et al. (1977) was published.

The description of a finite mixture model as a missing data problem was carried into the Bayesian literature. The amount of Bayesian analysis of mixtures was accelerated after the very influential papers by Tanner and Wong (1987) and Gelfand and Smith (1990). These two papers introduced into the statistical field two MCMC algorithms known as Data Augmentation and Gibbs sampling. The Gibbs algorithm is a special case of the more general Metropolis-Hastings algorithm detailed in Hastings (1970). These publications introduce methods that allow simulation from complex posterior distributions in a simple practical manner. For more details of these algorithms, see Sections 3.1.1 and 3.1.2. Also, Tierney (1994) and Gilks et al. (1996) provide a general background on the theory of MCMC methods. These methods allow for the sometimes very complex posterior distributions to be approximated adequately. A summary of some of the approximate Bayesian techniques used before the use of MCMC methods can be seen in Chapter 6 of Titterington et al. (1985). Two of the first papers that used MCMC methods in a mixture context are Diebolt and Robert (1990,

1994). They both use the MCMC methods, in the form of Data Augmentation and Gibbs Sampling, to estimate the posterior distributions for a k-component mixture of normals. Another of the early papers dealing with MCMC estimation techniques for mixture models is Escobar and West (1995). This paper is in a different setting to that of this thesis, because it is concerned with the Dirichlet Process Mixture Model. This model, detailed in Ferguson (1983) for the context of mixtures of normals, uses a Dirichlet process prior on a countably infinite set of mixture components. The analysis techniques used within this framework have many overlaps to the model structure used in this thesis, especially in the construction of the MCMC moves.

Most of the papers mentioned upon till now deal with the case of fitting the model where the number of components k is assumed known. However, the question, "How many components should be in the model?", is a very important one and, even though it has been researched for many years, a fully satisfactory solution is still to be found. Many different approaches have been proposed. An informal way of determining the number of components is through the use of graphical techniques, and a summary of some of these is given in Chapter 5 of Everitt and Hand (1981). The more formal hypothesis testing approach, of comparing two competing models, using for example the generalised likelihood ratio test, is not easily applied in the usual form because the standard regularity conditions for tests of this form do not hold - see Chapter 6 of MacLachlan and Peel (2000) for an example of how these regularity conditions break down. Other general model selection methods have been implemented including using Information Criteria such as Akaike's Information Criterion AIC, the Bayesian Information Criterion

BIC (Schwarz (1978)) and the Deviance Information Criterion DIC (Spiegelhalter et al. (2002)). The idea behind these is to fit the model that minimises the chosen criterion. The DIC is used by McGrory (2005) in her variational Bayes approximation framework. Her variational Bayes approximation method uses an iterative scheme to fit the model by essentially trying to approximate the true posterior density by a simpler density that is chosen to minimise the Kullback-Liebler divergence between these densities. Her simulation starting point has a high value of k and then elects to remove components throughout the simulation while recording the DIC. The simulation is run until the scheme converges to a particular value of k. The variational Bayes method has an advantage over most MCMC schemes in that it is less computationally intensive. This idea of using the Kullback-Liebler distance as a basis of testing between two competing models was first proposed in Mengersen and Robert (1996). Sahu and Cheng (2003) modifies the approach from Mengersen and Robert (1996) to help make the method more generally applicable, faster and easier to use in practice. Most of these papers do not use a prior on k to assess the number of components. Some other papers that also do not impose a prior on k are Carlin and Chib (1995), Chib (1995) and Raftery (1996). They all use a Bayes factor approach to test between models with k and $k+1$ components. The Bayes factor is the ratio of the marginal likelihoods under the two different models being compared. Another method of choosing the number of components is to construct the posterior of k. A prior distribution is placed on k and then estimates of the marginal likelihoods for each k are calculated. The posterior of k is then found by simple implementation of Bayes' theorem. This is an approach that was taken in Nobile (1994) and Roeder and Wasserman (1997). Nobile (2004) derives marginal likelihood

representations of these marginal likelihood terms required in the calculations.

A major development in this area of research came with the landmark paper of Green (1995). This important paper derived an MCMC scheme known as Reversible Jump MCMC (RJMCMC). This scheme allowed the MCMC sampler to jump between different models, meaning that in the case of finite mixture models the sampler can jump between models with differing numbers of components. This cross-model sampling method allows the sampler to sample from the joint posterior distribution of all the parameters including the number of components. The posterior distribution of k is then an easy quantity to calculate from the resulting MCMC output. The application of RJMCMC to finite mixture models was published in Richardson and Green (1997), where the authors detailed the framework for mixtures of univariate normal components. They evaluated the posterior distribution of k by calculating the relative frequency for each model visited throughout the simulation. Their model set-up contained all the parameters, namely the number of components, weights and component parameters, and therefore the jumps between different models were jumps between models of different dimensions. These dimension-jumping moves are usually called split/merge moves in the finite mixture context where a component is either split into two components or two components are merged together to create a single component. However, they are referred to as absorption/ejection type moves in the allocation sampler, see Chapter 3. A simpler reference for the RJMCMC theory contained in Green (1995) can be found in Chapter 6 of Green et al. (2003) under the title of "Trans-dimensional Markov chain Monte Carlo". For a more detailed summary of the model used in Richardson and Green (1997) see section

3.1.3. In RJMCMC the posterior distribution of k is simply estimated by the relative frequency that the sampler visits each model throughout the simulation. In RJMCMC the dimension-jumping moves become more computationally intensive when more and more parameters are contained in the model. A way of counteracting this significant increase in the number of parameters with an increase of k is to integrate some of the parameters out of the model, for example the component parameters and weights. This integration is computable in a closed form if conjugate priors are used for the parameters, but there are also cases when non-conjugate priors can be used, see Sections 2.4.3 and 2.4.4. Then the only unknowns left in the model are the number of components k and the vector of allocations conveying from which component each of the observations is coming. Authors who have worked within this framework are Nobile (1994), Casella et al. (2000), Steele et al. (2003), Fearnhead (2004) and Nobile and Fearnside (2007). The first author used this framework in his calculation of marginal likelihoods to construct the posterior of k. Casella et al. (2000) detail a partitioned importance sampling technique for use after integrating out the parameters and Steele et al. (2003) also propose an importance sampling technique to calculate what they call integrated likelihoods. Fearnhead (2004) presents this in a Dirichlet process model setting using particle filters, but integrating out only the component parameters. Nobile and Fearnside (2007) defines absorbtion/ejection moves for an MCMC sampler, which has both the component parameters and weights integrated out. They call this the *allocation sampler*. Full details of this method can be found in Chapter 3 of this thesis. This approach of integrating out the parameters has the other benefit that the state space of the MCMC sampler does not change with different component distributions or dimensions of the data. This

helps to make the analysis of multivariate data more practical. Analyses of multivariate data using RJMCMC for finite mixture models are few and far between because of the significant increase in computational work required. There have been some attempts though, including Zhang et al. (2004), Nobile and Fearnside (2007) and Dellaportas and Papageorgiou (2006), who all deal with multivariate normal components, and also Meligkotsidou (2007) using multivariate Poisson components for analysing multivariate count data.

There have been other methods proposed to enable cross-model sampling as alternatives to RJMCMC for Bayesian analysis of finite mixture models. Firstly, there was a jump diffusion approach suggested by Phillips and Smith (1996) which grew out of the methodology proposed by Grenander and Miller (1991, 1994). This paper defined a prior distribution over a set of models and their parameters. Furthermore, they then defined jump moves in an iterative jump-diffusion sampling scheme that allowed the sampler to jump between the models. The jumps are made at random times throughout the simulation and the diffusion moves sample the mixture parameters in between these jumps. They carefully define this method for a mixture of univariate normal components. Another alternative to the RJMCMC method is detailed in Stephens (1997a, 2000a). The method is based on continuous-time Markov birth-death processes as defined by Preston (1976). In this context, each mixture component is viewed as a point in the parameter space and then a point process is used to simulate from the posterior distribution. The birth-death aspect of the process allows jumps between mixture models of differing numbers of components by having births and deaths of mixture components. This method was compared to RJMCMC in

Cappé et al. (1995) and also showed the similarites between the two methodologies. Stephens has also published work on another important consideration when analysing mixtures, the *label switching problem*. This is a problem that arises when there is weak prior information discriminating between components. Furthermore, because the likelihood function of a finite mixture model of the form 1.1 is invariant with respect to a change in labels then there are essentially $k!$ modes in the posterior distribution. This obviously becomes a serious issue when parameter estimates and cluster analysis are being determined because the labels given to each component are not consistent throughout the simulation. This then leads to the posterior distributions of the parameters being symmetric. Hence, solutions to this problem have become a significant part of the research on finite mixture models in order that meaningful parameter estimates can be reported. Unfortunately, overcoming this label-switching problem can be a computationally expensive task. For a fuller summary and explanation of the problem and potential solutions to it see Section 3.2.5.

The next two chapters of this thesis set-up the finite mixture model and also an MCMC sampler to analyse this model tackling the problems of cross-model sampling, hyperparameter selection and label switching. This sampler is then tested in Chapter 4 using artificial data and then further in Chapter 5 using real datasets from analyses by previous authors.

Chapter 2

Model

This chapter introduces and defines the Bayesian finite mixture model that is used throughout the thesis.

Suppose $x = (x_1, \ldots, x_n)$ is a set of random variables (possibly vectors) that are independent and identically distributed and have a probability density function of the form

$$f(x|k, \lambda, \theta) = \sum_{j=1}^{k} \lambda_j q_j(x|\theta_j), \quad \lambda_j > 0, \quad \sum_{j=1}^{k} \lambda_j = 1. \tag{2.1}$$

where $\lambda = (\lambda_1, \ldots, \lambda_k)$ is the set of mixture weights which denote the probability that the random variable x_i follows one of the k possible distributions $q_j(x_i|\theta_j)$. These distributions are known as the mixture components. They belong to a known parametric family of distributions that are characterized by a set of parameters θ_j. For example, if a component was a normal distribution, then θ_j would contain the mean and variance. Furthermore, all the components are from the same specified family, thus forcing them to have the same functional form.

It should be noted that the meaning of the parameters changes when the number of components in the mixture changes. For example, the weight of the 1^{st} component can obviously have a different meaning and possibly different value as the number of components in the model increases. For ease of notation, this explicit dependence of the parameters on the dimension of the model has been be suppressed in the model notation.

2.1 Definition of Allocation vector, g

An analysis of the finite mixture model (2.1) can be interpreted as a missing-data problem. The missing-data is an indicator vector that denotes the component from which each observation has been generated. Let $g = (g_1, \ldots, g_n)$ be a latent allocation vector which denotes to which component each of the observations $x = (x_1, \ldots .x_n)$ is allocated. It will be called the allocation vector, although in some literature it is known as the membership vector. Then, obviously *a priori*, the x_i's arise from any of the k components $q_j(.)$ with probability λ_j.

Evaluation of the posterior distribution of g is a possible way of performing a cluster analysis on the data. A simple method of partitioning the observations into clusters, for a given k, is to allocate an observation to the component that has the highest estimated posterior probability. Binder (1978) reports on a Bayesian cluster analysis procedure for multivariate normal components, describing a method of estimating the allocation vector g and applying it to two examples. Multivariate normal mixtures are also used by Wruck et al. (2001) as a method for classification and discrimination. Some clustering methods and applications using mixtures models are given in MacLachlan and Basford (1987).

After tackling the label switching problem described in Section 3.2.5, one can calculate the posterior probability of an observation being allocated to a certain component, conditional on the number of components in the model. Also, one would be able to compute the posterior probability that two observations are allocated to the same component, See Section 2.5.4.

2.2 Bayesian model specification

The complete specification of the Bayesian model requires the bringing together of a prior distribution and a likelihood function. The prior is required to encompass all of the unknown variables (k, g, λ, θ). Thus, the full joint distribution of all the parameters and the data is

$$f(x, k, g, \lambda, \theta) = \pi(k)\pi(\lambda|k)f(g|k, \lambda)\pi(\theta|k, g, \lambda)f(x|k, g, \lambda, \theta). \tag{2.2}$$

Imposing further conditional independencies the joint distribution collapses to

$$f(x, k, g, \lambda, \theta) = \pi(k)\pi(\lambda|k)f(g|k, \lambda)\pi(\theta|k)f(x|k, g, \theta). \tag{2.3}$$

Note that throughout this thesis a prior or posterior distribution will be denoted by $\pi(.)$, a predictive distribution by $p(.)$ and any other distribution by $f(.)$. Careful consideration is required when specifying the priors in equation (2.3), so that the following assumption will hold.

An important characteristic of the approach to be discussed, is that the model is simplified by integrating out the mixture weights and component parameters.

Assumption 2.1 (Model Assumption). *Assume that*

$$\begin{aligned} f(x,k,g) &= \int f(x,k,g,\lambda,\theta)\mathrm{d}\lambda\mathrm{d}\theta \\ &= \pi(k)\int \pi(\lambda|k)f(g|k,\lambda)\mathrm{d}\lambda \int \pi(\theta|k)f(x|k,g,\theta)\mathrm{d}\theta \end{aligned} \tag{2.4}$$

can be computed in closed form.

This assumption is used by both Nobile (1994) and Steele et al. (2003) as a way of producing a method of analysis that changes little between different forms of the components and dimensions of the data. This is in contrast to standard models that leave the parameters in the model meaning that the complexity increases rapidly with an increase in dimension or change in component form.

The posterior from which one wishes to sample is $\pi(k, g|x)$ and is proportional to the joint distribution $f(x, k, g)$. Both the posterior on the number of components k, $\pi(k|x)$, and the posterior predictive distribution, $p(x_{n+1}|x)$, of a future observation can be evaluated easily from this posterior, see Sections 2.5.1 and 2.5.5. Also, with some extra effort, posterior distributions for the mixture weights (Section 2.5.2) and component parameters (Section 2.5.3) can be found, even though they have been integrated out of the model.

2.3 Prior distribution

2.3.1 Number of components, k

The prior on k has to be proper and have a support on the set of positive integers. It is critical that the prior is chosen wisely, because the posterior of k can be very

sensitive to the prior. One might first think of using a discrete uniform prior over the range $(1, \ldots, k_{max})$, where k_{max} is a computational upper bound on k. However, Nobile (2005) gave an argument for employing a prior proportional to a truncated $Poi(1)$ distribution. He justified using this prior by acknowledging the significant effect on the posterior distribution of k from models with empty components. Therefore, the idea of using a Poisson prior is to reduce this effect. The Poisson prior has been used by other authors, including Phillips and Smith (1996) and Stephens (2000a). A Poisson prior with a rate parameter equal to 1 has been used here and is therefore defined as

$$k \sim Poi(1) \quad \Leftrightarrow \quad \pi(k) \propto \tfrac{1}{k!} \quad k = 1, \ldots, k_{max}. \tag{2.5}$$

2.3.2 Mixture weights, λ

A popular choice for the prior on the mixture weights is the Dirichlet distribution, $Dir(\alpha_1, \ldots, \alpha_k)$ with $(\alpha_j > 0, j = 1, \ldots, k)$:

$$\pi(\lambda|k) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_k)} \lambda_1^{\alpha_1 - 1} \ldots \lambda_k^{\alpha_k - 1} \qquad \lambda_j > 0, \quad \sum_{j=1}^{k} \lambda_j = 1, \tag{2.6}$$

where $\alpha_0 = \sum_{j=1}^{k} \alpha_j$. We have chosen to use a symmetric Dirichlet distribution in this setting, where the hyperparameters are $\alpha_j = \alpha_1 = 1$. Consequently, the prior can be thought of as a uniform distribution on the simplex of the weights. This distribution is a conjugate prior for the mixture weight and is exactly the same as that used in Richardson and Green (1997) and Stephens (2000a).

2.3.3 Allocation vector, g

Assume the g_i's are conditionally independent given k and λ and that

$$\Pr[g_i = j|k, \lambda] = \lambda_j. \tag{2.7}$$

Following from the above assumptions

$$f(g|k, \lambda) = \prod_{j=1}^{k} \lambda_j^{n_j} \tag{2.8}$$

where n_j is the number of observations allocated by g to component j: $n_j = \text{card}\{A_j\}$ and $A_j = \{i : g_i = j\}$. Furthermore, conditional on g, the density of x_i can be given as $q_{g_i}(.|\theta_{g_i})$ and

$$f(x|k, \lambda, \theta, g) = \prod_{i=1}^{n} q_{g_i}(x_i|\theta_{g_i}). \tag{2.9}$$

Multiplying (2.9) together with (2.8), and integrating with respect to g, results in an expression for $f(x|k, \lambda, \theta)$ that equates to the finite mixture model defined in (2.1). For full details of this integration see page 23 of Nobile (1994).

For Assumption (2.1) to hold, the weights have to be integrated out of the expression $f(g|k, \lambda)f(\lambda|k)$. This produces

$$\begin{aligned} f(g|k) &= \int f(g|k, \lambda)\pi(\lambda|k)\mathrm{d}\lambda \\ &= \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_0 + n)} \prod_{j=1}^{k} \frac{\Gamma(\alpha_j + n_j)}{\Gamma(\alpha_j)}. \end{aligned} \tag{2.10}$$

Consequently, Assumption (2.1) is reduced to simply being able to integrate out

the component parameters from the remaining distributions in (2.3).

2.3.4 Component parameters, θ

Independent priors, conditional only on k and the hyperparameters ϕ, are chosen for θ,

$$\pi(\theta|k,\phi) = \prod_{j=1}^{k} \pi(\theta_j|\phi_j). \tag{2.11}$$

This prior has to be selected so that the parameters can be integrated out of the model. One way to ensure this, is to use a probability distribution from the exponential family of distributions as the component density, $q_j(x|\theta_j)$. Then, because the component density is a member of the exponential family, there exists a prior distribution conjugate to $q_j(x|\theta_j)$ that guarantees the component parameters can be integrated from the model. However, there are other probability distributions that are not members of the exponential family, but a prior distribution can still be defined in such a way that the component parameters can again be integrated out in closed form. Examples of both these situations can be seen in Section 2.4. These prior distributions have one assumption imposed on them.

Assumption 2.2. *Assume that the prior on θ_j does not change with a change in the number of components in the model.*

To understand this assumption, let the j^{th} component be a normal density, q_j, with a prior distribution on the mean and variance π_j, that has hyperparameters ϕ_j. Then, the j^{th} component will have these quantities when the model has $k = j, \ldots, k_{max}$ components in the model. A similar assumption is used in Nobile (2004).

2.3.5 Hyperparameters, ϕ

From the specification of the hyperparameters in the prior, two different cases arise for the model. Firstly, if all the component hyperparameters are assumed to be equal for all values of k, $(\alpha_j = \alpha_1, \phi_j = \phi_1)$, $j = 1, \ldots, k$, then the prior will be symmetric with respect to a permutation of the labels. This will be referred to as the symmetric case. If information distinguishing the components is known, then it should always be incorporated into the prior distribution. This would result in an asymmetric prior being defined and will therefore be calle dthe asymmetric case. Differences in the asymmetric hyperparameters lie in the vector ϕ, as the weight hyperparameters α are always all set equal to 1 in this framework, see Section 2.3.2. In the implementation of certain procedures, slight differences arise due to the symmetry or asymmetry of the prior distribution. These procedures will be discussed in more detail in Chapter 3. One of these procedures is the method of choosing the hyperparameter values, see Section 3.2.4.

2.3.6 Distribution of the data

Formally, we assume $x = (x_1, \ldots, x_n)$ is conditionally independent given the parameters (k, g, θ). Then

$$f(x|k, g, \theta) = \prod_{i=1}^{n} q_{g_i}(x_i|\theta_j) \tag{2.12}$$

and hence, with the prior on θ, (2.11), chosen to satisfy Assumption 2.1, it follows that

$$\begin{aligned} f(x|k,g,\phi) &= \int f(x|k,g,\theta)\pi(\theta|k,\phi)\mathrm{d}\theta \\ &= \int \prod_{i=1}^{n} q_{g_i}(x_i|\theta_j) \prod_{j=1}^{k} \pi(\theta_j|\phi_j)\mathrm{d}\theta \\ &= \prod_{j=1}^{k} \int \prod_{i\in A_j} q_j(x_i|\theta_j)\pi_j(\theta_j|\phi_j)\mathrm{d}\theta_j. \end{aligned} \tag{2.13}$$

Letting $p_j(x^j|\phi_j)$ denote the marginal density of the observations $x^j = \{x_i : i \in A_j\}$ allocated to component j, after integrating with respect to the prior of θ_j, one has

$$p_j(x^j|\phi_j) = \int \prod_{i\in A_j} q_j(x_i|\theta_j)\pi_j(\theta_j|\phi_j)\mathrm{d}\theta_j. \tag{2.14}$$

Note that if $A_j = \emptyset$ then $p_j(x^j|\phi_j) = 1$. Therefore, substituting (2.14) into (2.13), one has

$$f(x|k,g,\phi) = \prod_{j=1}^{k} p_j(x^j|\phi_j). \tag{2.15}$$

A full description of the Bayesian model (2.4) can be seen in Figure 2.1, comparing it to model (2.3), where the circles denote random parameters, and the squares denote fixed constants. These fixed constants can theoretically become random, if a hyperprior is put upon them. More comments on possible hyperpriors on ϕ can be seen in Section 3.2.4.

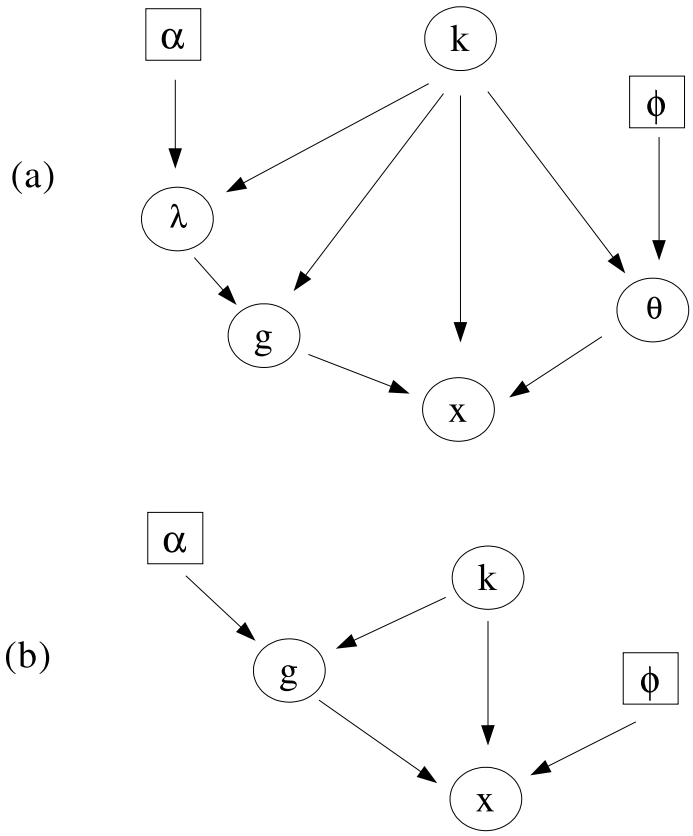

Figure 2.1: (a) is a directed acyclic graph (DAG) corresponding to the model (2.3) and (b) is a DAG corresponding to model (2.4) where the mixture weights and component parameters have been integrated out.

2.4 Examples

2.4.1 Mixtures of univariate normals

For illustrative purposes, one firstly looks at the simple case of a mixture of univariate normal components with unknown means and variances. Therefore, $q_{g_i}(x_i|\theta_{g_i})$ in (2.12) has the density $N(x_i|m_{g_i}, r_{g_i}^{-1})$ of a normal distribution with mean m_{g_i} and variance $r_{g_i}^{-1}$. The priors $\pi_j(\theta_j|\phi_j)$ in (2.11) are the usual conjugate priors for (m_j, r_j):

$$\begin{aligned} r_j &\sim Ga(\gamma_j, \delta_j) \\ m_j|r_j &\sim N(\mu_j, \{\tau_j r_j\}^{-1}), \end{aligned} \tag{2.16}$$

independently for each j, so that $\phi_j = (\mu_j, \tau_j, \gamma_j, \delta_j)$. Given k and g, the marginal distribution of the data allocated to the j-th component is given by

$$\begin{aligned} p_j(x^j|\phi_j) &= \pi^{-n_j/2}\left[\frac{\tau_j}{\tau_j+n_j}\right]^{1/2}\frac{\Gamma(\gamma_j+\{n_j/2\})}{\Gamma(\gamma_j)} \quad \cdot \\ &\qquad (2\delta_j)^{\gamma_j}\left\{2\delta_j+\sum_{i\in A_j}(x_i-\overline{x}_j)^2+\frac{\tau_j n_j}{\tau_j+n_j}(\overline{x}_j-\mu_j)^2\right\}^{-(\gamma_j+\{n_j/2\})}, \end{aligned}$$

where $\overline{x}_j = (1/n_j)\sum_{i\in A_j} x_i$. See page 21 of Nobile (1994) for more details of this result.

2.4.2 Mixtures of multivariate normals

Extending to the multivariate normal case with b-variate normal components yields $q_{g_i}(x_i|\theta_{g_i}) = N_b(x_i|m_{g_i}, r_{g_i}^{-1})$, where m_{g_i} is the mean vector, and r_{g_i} the precision matrix. The corresponding multivariate generalisations of the priors

used in the univariate case are

$$\begin{aligned} r_j &\sim W_b(\nu_j, \xi_j) \\ m_j | r_j &\sim N_b(\mu_j, \{\tau_j r_j\}^{-1}), \end{aligned} \tag{2.17}$$

independently for each j, where $W_b(\nu_j, \xi_j)$ is a b-variate Wishart distribution with ν_j degrees of freedom, and a precision matrix ξ_j. The remaining hyperparameters μ_j and τ_j are a b-vector and a positive real number. Therefore $\phi_j = \{\mu_j, \tau_j, \nu_j, \xi_j\}$ are the hyperparameters for this distribution. The marginal density of the data allocated to component j, given k and g, is

$$\begin{aligned} p_j(x^j|\phi_j) &= \pi^{-bn_j/2} \left[\frac{\tau_j}{\tau_j + n_j}\right]^{b/2} \prod_{s=1}^{b} \frac{\Gamma(\{\nu_j + n_j + 1 - s\}/2)}{\Gamma(\{\nu_j + 1 - s\}/2)} \; |\xi_j|^{\nu_j/2} \quad . \\ & \left| \xi_j + \sum_{i \in A_j} (x_i - \overline{x}_j)(x_i - \overline{x}_j)^\top + \frac{\tau_j n_j}{\tau_j + n_j} (\overline{x}_j - \mu_j)(\overline{x}_j - \mu_j)^\top \right|^{-(\nu_j + n_j)/2} \end{aligned}$$

where $\overline{x}_j$ is the sample mean vector of the observations allocated to component j. See page 42 of Nobile (1994) for more details of this result.

2.4.3 Mixtures of uniforms

Next, one looks at the case of the components following a uniform distribution. An example of a mixture of uniforms is the simple graphical tool, the histogram. This is a special case where all the distributions have equal length, and they also do not overlap. Therefore, each component is a uniform distribution over a single bin of the histogram. There is a significant drawback about the use of mixtures of uniforms, in that they can only be used for density estimation because of their lack of identifiability. This is noted on page 36 of Titterington et al. (1985), where

they state that a mixture of uniforms is not even identifiable up to a permutation of the labels. Hence, no parametric inference is meaningful in this case. Another problem that exists for the uniform distribution is that a conjugate prior is only avaiable for the $Unif(0, a)$ distribution. This conjugate prior distribution is the Pareto distribution. Therefore, since a $Unif(a, b)$ distribution is being used here the prior chosen for the component parameters $\{a, b\}$, is not conjugate. However, a prior can still be chosen in such a way that assumption 2.1 holds. Assume the components have the form $q_{g_i}(x_i|\theta_{g_i}) = Unif(x_i|a_{g_i}, b_{g_i})$, where (a_{g_i}, b_{g_i}) are the lower and upper bounds of the distribution. Then, let the parameters (a_j, b_j), $j = 1, \ldots, k$ be independent a priori with density

$$\pi_j(a_j, b_j|\phi_j) = 1/(2\phi_j^2), \quad -\phi_j < a_j < b_j < \phi_j \tag{2.18}$$

where $\phi_j = \phi$ is a positive constant. A point to note is that there should be no duplicate values in the data. This is only possible due to rounding data values because from simple measure theory no two data values will exactly coincide. The parameters are integrated out, yielding the marginal density of the data allocated to component j as

$$p_j(x^j|\phi) = \begin{cases} \frac{1}{2\phi^2} \frac{\left[(x^j_{(n_j)} - x^j_{(1)})^{-n_j+2} - (x^j_{(n_j)} + \phi)^{-n_j+2} - (\phi - x^j_{(1)})^{-n_j+2} + (2\phi)^{-n_j+2}\right]}{(n_j-1)(n_j-2)} & n_j > 2 \\ \frac{1}{2\phi^2}\left[\log \frac{(\phi - x^j_{(1)})(\phi + x^j_{(2)})}{2\phi(x^j_{(2)} - x^j_{(1)})}\right] & n_j = 2 \\ \frac{1}{2\phi^2}\left[x^j_{(1)} \log \frac{\phi - x^j_{(1)}}{\phi + x^j_{(1)}} + \phi \log \frac{(2\phi)^2}{(\phi - x^j_{(1)})(\phi + x^j_{(1)})}\right] & n_j = 1 \end{cases} \tag{2.19}$$

where $x^j_{(i)}$ denotes the i-th order statistic of x^j. For the full details of the derivation of (2.19), see Appendix (A.1).

2.4.4 Mixtures of sign-shifted-exponentials

Mixtures of exponential distributions have been analysed by Gruet et al. (1999) using a RJMCMC scheme. However, to have a more general class of distributions, the sign-shifted exponential distribution is defined here. This is a three-parameter distribution. Firstly, there is the usual rate parameter, ω, for an exponential distribution. Then, one also defines a shift parameter, a, to allow the distribution to move along the x-axis and finally, a sign parameter to characterise the direction of the exponential decay.

If one considers x to be an observation from a sign-shifted exponential distribution, then x would have a density of the form

$$f(x|\theta) = \begin{cases} \omega e^{-\omega(x-a)} I_{(a,\infty)}(x), & s = 1 \\ \omega e^{\omega(x-a)} I_{(-\infty,a)}(x), & s = -1 \end{cases} \tag{2.20}$$

where s takes values from the set $S = \{-1, 1, \}$, $a \in \mathbb{R}$ and $\omega > 0$. Hence, for this distribution the parameter $\theta = \{s, a, \omega\}$. For notation, $SSExp(s, a, \omega)$ will be used to denote a sign-shifted exponential distribution.

Then, a sign-shifted exponential component q_j containing n_j observations, x^j, has a distribution given by

$$q_j(x^j|\theta_j) = \begin{cases} \prod_{i=1}^{n_j} \left[\omega_j e^{-\omega_j(x^j_i - a_j)} I_{(a_j,\infty)}(x^j_i)\right], & s_j = 1 \\ \prod_{i=1}^{n_j} \left[\omega_j e^{\omega_j(x^j_i - a_j)} I_{(-\infty,a_j)}(x^j_i)\right], & s_j = -1, \end{cases} \tag{2.21}$$

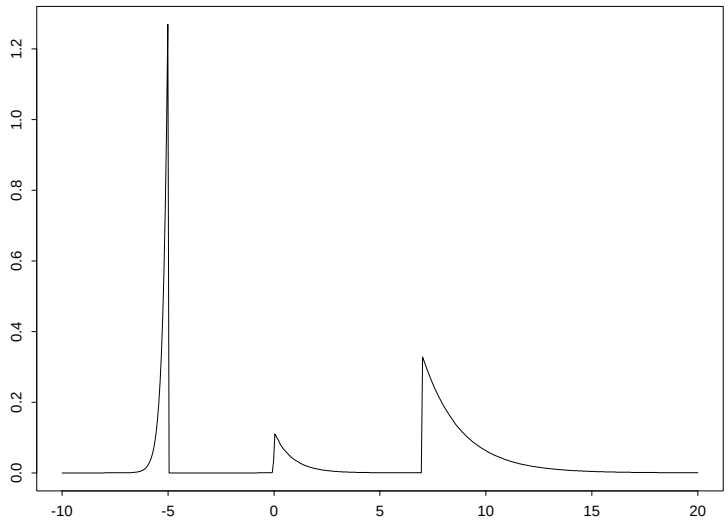

Figure 2.2: Example of a Sign-Shifted-Exponential density. Shown is a 3 component mixture of Sign-Shifted-Exponentials, where the parameters in the model are as follows, $\lambda = (\frac{3}{5}, \frac{3}{10}, \frac{1}{10})$, $s = (1, -1, 1)$, $a = (7, -5, 0)$ and $\omega = (\frac{1}{2}, 4, 1)$.

where $\theta_j = (\omega_j, a_j, s_j)$. Here s_j corresponds to the sign parameter, a_j to the shift parameter and ω_j to the rate parameter of the j^{th} component in the mixture. The prior imposed on the rate parameter, ω_j, is a $Ga(\kappa_j, \beta_j)$ where κ_j and β_j are known positive values:

$$\pi(\omega_j) = \frac{\beta_j^{\kappa_j} \omega_j^{\kappa_j - 1} e^{-\beta_j \omega_j}}{\Gamma(\kappa_j)}, \qquad \kappa_j, \beta_j > 0. \tag{2.22}$$

The prior on the shift parameter a_j, given the rate parameter ω_j, is $Laplace(0, \gamma_j \omega_j)$:

$$\pi(a_j|\omega_j) = \frac{\gamma_j \omega_j}{2} e^{-\gamma_j \omega_j |a_j|} \tag{2.23}$$

Finally, let

$$\pi(s_j) = \begin{cases} \rho_j, & s = 1 \\ (1-\rho_j), & s = -1 \end{cases} \tag{2.24}$$

where ρ_j is $0 < \rho_j < 1$. For this distribution, $\phi_j = (\kappa_j, \beta_j, \gamma_j, \rho_j)$. The marginal density of the data allocated to component j, given k and g, is given by

$$p_j(x^j|\phi_j) = \rho_j p_j(x^j|s_j = 1) + (1-\rho_j)p_j(-x^j|s_j = 1), \tag{2.25}$$

which uses the fact that $p_j(-x^j|s_j = 1) = p_j(x^j|s_j = -1)$, see equation (A.20) in Appendix A.2. There are two cases that arise when calculating the distribution $p_j(x^j|s_j = 1)$. Firstly, when $n_j \neq \gamma_j$,

$$\begin{aligned} p_j(x^j|s_j = 1) = {} & \frac{\gamma_j \beta_j^{\kappa_j} \Gamma(\kappa_j + n_j)}{2\Gamma(\kappa_j)} \cdot \\ & \Bigg[\Big(n_j + \gamma_j\Big)^{-1} \Big(\beta_j + \sum x^j - (n_j + \gamma_j)\min(0, x^j_{(1)})\Big)^{-(\kappa_j + n_j)} \\ & + \Big(n_j - \gamma_j\Big)^{-1} \Big(\beta_j + \sum x^j - (n_j - \gamma_j)\max(0, x^j_{(1)})\Big)^{-(\kappa_j + n_j)} \\ & - \Big(n_j - \gamma_j\Big)^{-1} \Big(\beta_j + \sum x^j\Big)^{-(\kappa_j + n_j)} \Bigg], \end{aligned} \tag{2.26}$$

where $x^j_{(i)}$ denoting the i-th order statistic of x^j and $\sum x^j = \sum_{i=1}^{n_j} x^j_i$. Then, secondly, the case where $n_j = \gamma_j$,

$$p_j(x^j|s_j = 1) = \frac{\gamma_j \beta_j^{\kappa_j}}{2\Gamma(\kappa_j)} \cdot \left[\Gamma(\kappa_j + n_j)\Big(n_j + \gamma_j\Big)^{-1}\left(\beta_j + \sum x^j - (n_j + \gamma_j)\min(0, x^j_{(1)})\right)^{-(\kappa_j + n_j)} + \max(0, x^j_{(1)})\Gamma(\kappa_j + n_j + 1)\left(\beta_j + \sum x^j\right)^{-(\kappa_j + n_j + 1)}\right] \tag{2.27}$$

See Appendix (A.2) for the full details of the derivations of these two distributions.

2.5 Posterior distributions

Finite mixture models can be summarised by looking at the posterior distributions of all the parameters in the model. Even though the parameters λ and θ do not explicitly appear in the sampling procedure, the posterior distributions can be calculated using the MCMC output. This is in contrast to the usual RJMCMC scheme defined in Richardson and Green (1997), where all the parameters explicitly appear in the MCMC sampling procedure.

There are five posterior distributions that may be of interest:

- Posterior distribution of the number of components, k
- Posterior distribution of the component weights, λ
- Posterior distribution of component parameters, θ
- Posterior distribution of allocation vector, g

- Posterior predictive distribution of a future observation, x_{n+1}.

The posterior distributions for the weights, component parameters and the allocation vector are all calculated conditional on the number of components in the model. Summarising the features of a finite mixture model using their parameters can be difficult. The main reason for this is that the number of parameters in the mixture can be very large for even fairly simple mixture distributions. However, they still play an important role as they are needed in order to evaulate the posterior predictive distributions of future observables.

Note that when in the symmetric case, the problem of "label switching" should be tackled first to obtain more meaningful posterior results, since attempting to do parametric inference without this would cause the posteriors for all components to be symmetric. For details on how this problem is overcome see Section 3.2.5.

2.5.1 Posterior distribution of the number of components

This is the simplest posterior to evaluate since it is a byproduct of the MCMC sampler. A record of the changing states of k throughout the simulation is kept in order to estimate this posterior. The posterior probability for having k components in the model is found by looking at the frequency of the sampler being in a state with k components. Thus, a Monte Carlo estimate of the posterior probability of having k_0 components in the model can be defined as

$$\widehat{\pi(k_0|x)} = \frac{\sum_{m=1}^{N} I(k^{(m)} = k_0)}{N}, \tag{2.28}$$

where $k^{(m)}$ is the number of components in the m^{th} simulation, N is the total number of allocation vectors simulated by the MCMC sampler, and I is the indicator function.

2.5.2 Posterior distributions of component weights

We obtain

$$\lambda|k, g, x \sim Dir(\alpha'_1, \ldots, \alpha'_k). \tag{2.29}$$

where $\alpha'_j = \alpha_j + n_j$. Therefore, the marginal posterior distribution of the weights unconditional on g are found by averaging (2.29) over the posterior distribution of g,

$$\lambda|k, x \sim \sum_g \pi(g|k, x) Dir(\alpha'_1, \ldots, \alpha'_k) \tag{2.30}$$

Note that it only makes sense to calculate the posterior distribution of the weights given a certain value of k, because the meaning of the weight for component j changes as k changes.

2.5.3 Posterior distributions of component parameters

For the component parameters, θ, one has

$$\pi(\theta|k, g, x, \phi) = \prod_{j=1}^{k} \pi_j(\theta_j|x^j, \phi_j), \tag{2.31}$$

where $\pi_j(\theta_j|x^j,\phi_j)$ denotes the posterior of θ_j given that g allocates x^j to component j. In general, this distribution can be written as

$$\pi_j(\theta_j|x^j,\phi_j) = \frac{\pi_j(\theta_j|\phi_j)\prod_{i\in A_j} q_j(x_i|\theta_j)}{p_j(x^j|\phi_j)}, \tag{2.32}$$

where the normalizing constants $p_j(x^j|\phi_j)$ were defined in (2.14). If conjugate priors $\pi_j(\theta_j|\phi_j)$ are used, the factors in (2.31) take the simple form $\pi_j(\theta_j|x^j,\phi_j) = \pi_j(\theta_j|\phi'_j)$, where ϕ'_j is the updated value of the hyperparameter, according to the relevant rule for the family of distributions in question. Thus

$$\pi(\theta|k,x,\phi) = \sum_g \pi(g|k,x)\prod_{j=1}^{k}\pi_j(\theta_j|x^j,\phi_j). \tag{2.33}$$

2.5.3.1 Mixtures of univariate normals

For univariate normal components and the prior described in Section 2.4.1, the posteriors $\pi_j(\theta_j|x^j,\phi_j)$ in (2.31) are as follows: $r_j|g,x \sim Ga(\gamma'_j,\delta'_j)$ and $m_j|r_j,g,x \sim N(\mu'_j,\{\tau'_j r_j\}^{-1})$, where the updated hyperparameter values are

$$\gamma'_j = \gamma_j + \frac{n_j}{2}, \qquad \delta'_j = \delta_j + \frac{1}{2}\sum_{i\in A_j}(x_i^j - \overline{x}_j)^2 + \frac{\tau_j n_j}{2(\tau_j+n_j)}(\overline{x}_j - \mu_j)^2,$$

$$\tau'_j = \tau_j + n_j, \qquad \mu'_j = \frac{\tau_j\mu_j + n_j\overline{x}_j}{\tau_j+n_j}.$$

See page 169 of DeGroot (1970) for the derivation of these updated hyperparameter values.

2.5.3.2 Mixtures of multivariate normals

For multivariate normal components with the prior discussed in Section 2.4.2, the posteriors $\pi_j(\theta_j|x^j,\phi_j)$ are $r_j|g,x^j \sim W_b(\nu'_j,\xi'_j)$ and $m_j|r_j,g,x^j \sim N_b(\mu'_j,\{\tau'_j r_j\}^{-1})$, where

$$\xi'_j = \xi_j + \sum_{i\in A_j}(x_i^j - \overline{x}_j)(x_i^j - \overline{x}_j)^\top + \frac{\tau_j n_j}{\tau_j + n_j}(\mu_j - \overline{x}_j)(\mu_j - \overline{x}_j)^\top,$$

$$\nu'_j = \nu_j + n_j, \qquad \tau'_j = \tau_j + n_j, \qquad \mu'_j = \frac{\tau_j\mu_j + n_j\overline{x}_j}{\tau_j + n_j}.$$

See page 178 of DeGroot (1970) for the derivation of these updated hyperparameter values.

2.5.3.3 Mixtures of uniforms

The posterior distributions of the parameters for uniform components need to be derived, even though, as discussed in Section 2.4.3, inference about the parameters in this case is meaningless, due to the lack of identifiability feature of these models. For uniform components with the prior discussed in Section 2.4.3, the posteriors have the form

$$\pi_j(\theta_j|x^j,\phi_j) = \frac{1}{p_j(x^j|\phi)}\frac{1}{2\phi^2(b_j-a_j)^{n_j}}, \qquad -\phi < a_j < x^j_{(1)}, \quad x^j_{(n_j)} < b_j < \phi \tag{2.34}$$

where $p_j(x^j|\phi)$ is given in equation (2.19).

2.5.3.4 Mixtures of sign-shifted exponentials

For sign-shifted exponential components the posterior density functions for the shift and rate parameters are calculated conditional on the sign of the component. Therefore, the first posterior density to be evaluated is for the sign parameter, s. The posterior probability that $s_j = 1$ is

$$p_j(s_j = 1|x^j) = \frac{\rho_j p_j(x^j|s_j = 1)}{p_j(x^j|\phi_j)} \tag{2.35}$$

and for $s_j = -1$

$$p_j(s_j = -1|x^j) = \frac{(1-\rho_j) p_j(x^j|s_j = -1)}{p_j(x^j|\phi_j)} \tag{2.36}$$

where $p_j(x^j|\phi_j)$ is defined by (2.25). The quantities $p_j(x^j|s_j = 1)$ and $p_j(x^j|s_j = -1)$ in (2.35) and (2.36) are both calculated from (2.26) and (2.27).

The posterior densities for the remaining parameters can be written as proportionalities where the proportionality constant is simply $p_j(x^j|\phi_j)$. Hence, the posterior density for the rate parameter, ω, conditional on $s = 1$ has the following form if $n_j \neq \gamma_j$:

$$\begin{aligned}
&\pi(\omega_j|k, g, x, s_j = 1) \propto \\
&\quad \frac{\gamma_j \beta_j^{\kappa_j}}{2\Gamma(\kappa_j)} \omega_j^{\kappa_j + n_j} \exp\left\{-\omega_j(\beta_j + \sum x_i^j)\right\} \cdot \\
&\quad \left[\frac{\exp\left\{\omega_j(n_j + \gamma_j)\min(0, x_{(1)})\right\}}{\omega_j(n_j + \gamma_j)} + \frac{\exp\left\{\omega_j(n_j - \gamma_j)\max(0, x_{(1)})\right\} - 1}{\omega_j(n_j - \gamma_j)}\right].
\end{aligned} \tag{2.37}$$

If $n_j = \gamma_j$, one has

$$\begin{aligned}\pi(\omega_j|k,g,x,s_j=1) \propto \\ \frac{\gamma_j\beta_j^{\kappa_j}}{2\Gamma(\kappa_j)}\left[\frac{1}{n_j+\gamma_j}\omega_j^{\kappa_j+n_j-1}\exp\left\{-\omega_j(\beta_j+\sum x_i^j-(n_j+\gamma_j)\min(0,x_{(1)})\right\}\right. \\ \left.+\max(0,x_{(1)})\omega^{\kappa_j+n_j}\exp\left\{-\omega_j(\beta_j+\sum x_i^j)\right\}\right].\end{aligned} \tag{2.38}$$

These two results, (2.37) and (2.38), are a by-product of the derivation of $p_j(x^j|s_j = 1)$, see Appendix A.2. The posterior densities for the rate parameter when $s = -1$ are of a similar form to (2.37) and (2.38). Hence, when $n_j \neq \gamma_j$,

$$\begin{aligned}\pi(\omega_j|k,g,x,s_j=-1) \propto \\ \frac{\gamma_j\beta_j^{\kappa_j}}{2\Gamma(\kappa_j)}\omega_j^{\kappa_j+n_j}\exp\left\{-\omega_j(\beta_j-\sum x_i^j)\right\}\cdot \\ \left[\frac{\exp\left\{\omega_j(n_j+\gamma_j)\min(0,-x_{(n)})\right\}}{\omega_j(n_j+\gamma_j)}+\frac{\exp\left\{\omega_j(n_j-\gamma_j)\max(0,-x_{(n)})\right\}-1}{\omega_j(n_j-\gamma_j)}\right],\end{aligned} \tag{2.39}$$

and if $n_j = \gamma_j$

$$\begin{aligned}\pi(\omega_j|k,g,x,s_j=-1) \propto \\ \frac{\gamma_j\beta_j^{\kappa_j}}{2\Gamma(\kappa_j)}\left[\frac{1}{n_j+\gamma_j}\omega_j^{\kappa_j+n_j-1}\exp\left\{-\omega_j(\beta_j-\sum x_i^j-(n_j+\gamma_j)\min(0,-x_{(n)})\right\}\right. \\ \left.+\max(0,-x_{(n)})\omega^{\kappa_j+n_j}\exp\left\{-\omega_j(\beta_j-\sum x_i^j)\right\}\right].\end{aligned} \tag{2.40}$$

The posterior densities for the shift parameters, conditional on s, are found by integrating the rate parameters out of $p_j(x^j|a_j, \omega_j, s_j = 1, \phi_j)$ and $p_j(x^j|a_j, \omega_j, s_j = -1, \phi_j)$. The integral for $s = 1$ is defined as follows:

$$\begin{aligned}
&\pi(a_j|k, g, x, \phi_j, s_j = 1) \\
&\quad \propto \int_0^\infty \omega_j^{n_j} \exp\left\{-\omega_j \left(\sum x^j - n_j a_j\right)\right\} \frac{\gamma_j \beta_j^{\kappa_j} \omega_j^{\kappa_j} \exp\{-\omega_j(\beta_j + \gamma_j|a_j|)\}}{2\Gamma(\kappa_j)} \mathrm{d}\omega_j \\
&\quad = \frac{\gamma_j \beta_j^{\kappa_j}}{2\Gamma(\kappa_j)} \int_0^\infty \omega_j^{\kappa_j + n_j} \exp\left\{-\omega_j \left(\beta_j + \sum x^j - (n_j - \gamma_j) a_j\right)\right\} \mathrm{d}\omega_j \\
&\qquad + \int_0^\infty \omega_j^{\kappa_j + n_j} \exp\left\{-\omega_j \left(\beta_j + \sum x^j - (n_j + \gamma_j) a_j\right)\right\} \mathrm{d}\omega_j \\
&\quad = \frac{\gamma_j \beta_j^{\kappa_j}}{2\Gamma(\kappa_j)} \left[\frac{\Gamma(\kappa_j + n_j + 1)}{(\beta_j + \sum x^j - (n_j - \gamma_j) a_j)^{\kappa_j + n_j + 1}} + \frac{\Gamma(\kappa_j + n_j + 1)}{(\beta_j + \sum x^j - (n_j + \gamma_j) a_j)^{\kappa_j + n_j + 1}} \right].
\end{aligned} \tag{2.41}$$

A similar integral is calculated for $s = -1$ resulting in a density of the same form as (2.41):

$$\begin{aligned}
&\pi(a_j|k, g, x, \phi_j, s_j = -1) \\
&\quad \propto \frac{\gamma_j \beta_j^{\kappa_j}}{2\Gamma(\kappa_j)} \left[\frac{\Gamma(\kappa_j + n_j + 1)}{(\beta_j - \sum x^j + (n_j - \gamma_j) a_j)^{\kappa_j + n_j + 1}} + \frac{\Gamma(\kappa_j + n_j + 1)}{(\beta_j - \sum x^j + (n_j + \gamma_j) a_j)^{\kappa_j + n_j + 1}} \right].
\end{aligned} \tag{2.42}$$

2.5.4 Posterior distribution of allocation vector

Analysing the posterior distribution of the allocation vector to find out what observations can be grouped together gives a cluster analysis for the data. This posterior distribution can be summarised in a number of different ways, two of which are defined here. Firstly, a Monte Carlo estimate can be evaluated for the posterior probability that an observation g_i is allocated to a specific component, after the label-switching problem has been overcome, conditional on a certain value k, by

$$\widehat{\pi}(g_i = j|k, x) = \frac{\sum_{m=1}^{N} I(g_i^{(m)} = j)}{N^k} \qquad i = 1, \cdots, n \quad j = 1, \cdots, k \tag{2.43}$$

where $g^{(m)}$ is the m^{th} allocation vector produced by the allocation sampler, N is the total number of allocation vectors, N^k is the number of allocation vectors with k components and I is the usual indicator function. Secondly, another quantity that might be of interest is a Monte Carlo estimate for the posterior probability that two observations are allocated to the same component unconditional on k. The probability that observation g_i is allocated to the same component as observation g_j is calculated as follows from the N allocation vectors simulated using the allocation sampler:

$$\widehat{\pi}(g_i = g_j|x) = \frac{\sum_{m=1}^{N} I(g_i^{(m)} = g_j^{(m)})}{N} \qquad i = 1, \cdots, n \quad j = 1, \cdots, n. \tag{2.44}$$

2.5.5 Posterior predictive distribution

The posterior predictive distribution is of great importance when using mixtures as a density estimation tool. Assume that the future observation x_{n+1}, conditional on all the parameters (k, λ, g, θ), is independent of the previous data x. Then x_{n+1} has a distribution of the same form as (2.1):

$$f(x_{n+1}|k, \lambda, \theta, g, x, \phi) = \sum_{j=1}^{k} \lambda_j q_j(x_{n+1}|\theta_j). \tag{2.45}$$

A point to note is that this distribution is not conditional on g_{n+1}, i.e. we do not specify the component to which the new observation x_{n+1} belongs. Like (2.1), (2.45) has to be integrated with respect to the joint distribution of λ and θ given k, g and x as follows:

$$\begin{aligned} f(x_{n+1}|k, g, x, \phi) &= \int\int f(x_{n+1}|k, \lambda, \theta) f(\theta|k, \lambda, g, x, \phi) f(\lambda|k, g, x) \mathrm{d}\theta \mathrm{d}\lambda \\ &= \int \left[\int f(x_{n+1}|k, \lambda, \theta) f(\theta|k, \lambda, g, x, \phi) \mathrm{d}\theta \right] f(\lambda|k, g, x) \mathrm{d}\lambda, \end{aligned}$$

where

$$\begin{aligned} \int f(x_{n+1}|k, \lambda, \theta) f(\theta|k, \lambda, g, x, \phi) \mathrm{d}\theta &= \int \sum_{j=1}^{k} \lambda_j q_j(x_{n+1}|\theta_j) \pi_j(\theta_j|x^j, \phi_j) \mathrm{d}\theta_j \\ &= \sum_{j=1}^{k} \lambda_j \int q_j(x_{n+1}|\theta_j) \pi_j(\theta_j|x^j, \phi_j) \mathrm{d}\theta_j \\ &= \sum_{j=1}^{k} \lambda_j p_j(x_{n+1}|x^j, \phi_j) \end{aligned} \tag{2.46}$$

with $p_j(x_{n+1}|x^j, \phi_j)$ being the posterior predictive density corresponding to component j.

Thus, using (2.29), the posterior predictive distribution conditional on k and g becomes

$$\begin{aligned} f(x_{n+1}|k,g,x,\phi) &= \int \sum_{j=1}^{k} \lambda_j p_j(x_{n+1}|x^j,\phi_j) f(\lambda|k,g,x)\mathrm{d}\lambda \\ &= \int \sum_{j=1}^{k} p_j(x_{n+1}|x^j,\phi_j) \frac{\Gamma(\alpha_0')}{\Gamma(\alpha_1')\ldots\Gamma(\alpha_k')} \lambda_1^{\alpha_1'-1}\ldots\lambda_k^{\alpha_k'-1}\lambda_j \mathrm{d}\lambda \\ &= \sum_{j=1}^{k} p_j(x_{n+1}|x^j,\phi_j) \frac{\Gamma(\alpha_0')}{\Gamma(\alpha_1')\ldots\Gamma(\alpha_k')} \int \lambda_1^{\alpha_1'-1}\ldots\lambda_k^{\alpha_k'-1}\lambda_j \mathrm{d}\lambda \\ &= \sum_{j=1}^{k} p_j(x_{n+1}|x^j,\phi_j) \frac{\Gamma(\alpha_0')}{\Gamma(\alpha_1')\ldots\Gamma(\alpha_k')} \frac{\Gamma(\alpha_1')\ldots\Gamma(\alpha_j'+1)\ldots\Gamma(\alpha_k')}{\Gamma(\alpha_0'+1)} \\ &= \sum_{j=1}^{k} \frac{\alpha_j'}{\alpha_0'} p_j(x_{n+1}|x^j,\phi_j). \end{aligned} \tag{2.47}$$

If (2.47) is averaged over the joint distribution of k and g, this produces a posterior predictive distribution of the form

$$p(x_{n+1}|x,\phi) = \sum_{k,g} f(k,g|x,\phi) \sum_{j=1}^{k} \frac{\alpha_j'}{\alpha_0'} p_j(x_{n+1}|x^j,\phi_j). \tag{2.48}$$

Furthermore, it is easy to evaluate the posterior predictive conditioned on a certain value of k, by only averaging over the g vectors corresponding to that k:

$$p(x_{n+1}|k,x,\phi) = \sum_{g} f(g|k,x,\phi) \sum_{j=1}^{k} \frac{\alpha_j'}{\alpha_0'} p_j(x_{n+1}|x^j,\phi_j). \tag{2.49}$$

Some simplifications are available for $p_j(x_{n+1}|x^j, \phi_j)$'s. Firstly, if the priors

on the component parameters are conjugate, as in the univariate or multivariate normal component cases, then $\pi(\theta_j|x^j,\phi_j)$ in (2.46) can be written as $\pi(\theta_j|\phi'_j)$, where ϕ'_j contains the updated hyperparameter values. Also, the posterior predictive distributions $p_j(x_{n+1}|x^j,\phi_j)$ can be simplified by substituting (2.32) into their definition in equation (2.46):

$$\begin{aligned} p_j(x_{n+1}|x^j,\phi_j) &= \int q_j(x_{n+1}|\theta_j)\pi_j(\theta_j|x^j,\phi_j)\mathrm{d}\theta_j \\ &= \int q_j(x_{n+1}|\theta_j)\frac{\pi_j(\theta_j|\phi_j)\prod_{i\in A_j} q_j(x_i|\theta_j)}{p_j(x^j|\phi_j)}\mathrm{d}\theta_j \\ &= \frac{1}{p_j(x^j|\phi_j)}\int \prod_{i\in A_j\cup\{n+1\}} q_j(x_i|\theta_j)\pi(\theta_j|\phi_j)\mathrm{d}\theta_j \\ &= \frac{p_j(\tilde{x}^j|\phi_j)}{p_j(x^j|\phi_j)} \end{aligned} \tag{2.50}$$

where $\tilde{x}^j$ denotes the vector x^j augmented with x_{n+1}:$\{x_i : i \in A_j \cup \{n+1\}\}$. This version is used when calculating the posterior predictive distributions in the no-conjugate cases of uniform or sign-shifted exponential components.

2.5.5.1 Mixtures of univariate normals

The posterior predictive density $p_j(x_{n+1}|x^j,\phi_j)$ is the density of a univariate t distribution with $2\gamma'_j$ degrees of freedom, location μ'_j and precision $\{\gamma'_j/\delta'_j\}\{\tau'_j/(1+\tau'_j)\}$.

2.5.5.2 Mixtures of multivariate normals

The posterior predictives $p_j(x_{n+1}|x^j,\phi_j)$ are b-variate t distributions with $\nu'_j - b + 1$ degrees of freedom, location vectors μ'_j and precision matrix $\{\tau'_j/(1+\tau'_j)\}(\nu'_j - b + 1)\xi'^{-1}_j$.

2.5.5.3 Mixtures of uniforms

The posterior predictives $p_j(x_{n+1}|x^j, \phi_j)$ are found by using expression (2.50) because the priors on the parameters are not conjugate. The expression (2.19) is then used to calculate $p_j(\tilde{x}^j|\phi_j)$ and $p_j(x^j|\phi_j)$ as required in (2.50).

2.5.5.4 Mixtures of sign-shifted-exponentials

The posterior predictives $p_j(x_{n+1}|x^j, \phi_j)$ are again found by using (2.50) because of the lack of conjugacy in the choice of prior distribution. The expression (2.25) is then used to calculate the relevant quantities in (2.50).

Chapter 3

The Allocation Sampler

This chapter introduces a Markov chain Monte Carlo (MCMC) sampler that explores the joint posterior distribution of k and g,

$$\pi(k, g|x, \phi) \propto f(k, g, x|\phi) = \pi(k)f(g|k)f(x|k, g, \phi), \tag{3.1}$$

where $\pi(k)$, $f(g|k)$ and $f(x|k, g, \phi)$ are given by expressions (2.5), (2.10) and (2.15). This sampler is given the name *the allocation sampler* because the sampler is essentially sampling allocation vectors, allowing the number of components in the model to change.

3.1 Markov chain Monte Carlo

Markov chain Monte Carlo methods can be traced as far back as the papers Metropolis et al. (1953) and Hastings (1970). However, it is only in the last two decades that these methods have started to be used to their full potential in statistics because of the large amount of computer power required to implement

these algorithms. The idea of MCMC is to use Monte Carlo averages of a Markov chain to approximate integrals that are essentially impossible to evaluate analytically. See Andrieu et al. (2003) for a simple introduction to MCMC methods from which a short summary is now given.

Suppose we define a random variable X, at time t, as X_t, with a finite state space S. Then a Markov chain $(X_0, \ldots, X_n)$ can be defined as a sequence of random variables generated by a Markov process in discrete time. This is a process in which the transition probabilities between different points in the state space S, are determined only by the random variable's current state, and not any previous state. These transition probabilities between states make up the transition matrix. In MCMC the Markov chain is required to possess certain properties, so that at equilibrium the observations from the chain are hopefully realisations from a target distribution π, in our case a posterior distribution. For the Markov chain, described by a particular transition matrix, to have a stationary distribution, it is sufficient for the chain to be irreducible and aperiodic. An irreducible chain is one in which all the possible states of the chain can communicate with each other. A Markov chain is defined to be aperiodic if the number of steps required to move between two states is not fixed to be a multiple of some integer. Another condition, which is used to ensure that the Markov chain has a stationary distribution π_i, is that of detailed balance. This condition can be defined as follows:

$$P(i \to j)\pi_i = P(j \to i)\pi_j, \tag{3.2}$$

where $P(i \to j)$ is the probability of moving from state i to state j, and π_j is the

target distribution value at state j. If this condition holds, the Markov chain is said to be reversible.

Extending this theory to cover continuous state spaces means the transition matrix becomes a transition kernel. Then, the problem to be answered in MCMC is how to define this so-called transition kernel, in such a way that the Markov chain eventually produces samples from the desired stationary distribution. This is a large area of research in statistics with new and modified algorithms for producing transition kernels, and hence Markov chains with the relevant stationary distributions, are being published continually. If a suitable transition kernel is found, then a Markov chain could be started from an initial value, say X_0. It would then be allowed to run according to the transition kernel for some time, in order for the chain to converge to the target distribution. This is what is termed a burn-in period B, and the sequence of the chain after the burn-in $(X_{B+1}, \ldots, X_{B+n})$ would be used for inference on the target distribution.

Two of the most popular algorithms used to create MCMC samplers are the Metropolis-Hastings algorithm and the Gibbs sampler.

3.1.1 Gibbs Sampling

The Gibbs sampler was first introduced by that name in the paper by Geman and Geman (1984) in the context of image analysis, but became more popular among statisticians after the papers by Tanner and Wong (1987) and Gelfand and Smith (1990). The main idea behind the sampler is to construct a Markov chain which converges to the target distribution, by considering only univariate conditional distributions. Obviously, sampling from univariate conditional distributions is

much easier than trying to sample directly from the full joint distribution.

The easiest way to understand how this method works is to look at the simple case of a bivariate random variable, say (X, Y). Now, suppose we wish to make inference about the marginal distributions of this random variable, namely $f(X)$ and $f(Y)$. Then, the Gibbs sampler only requires knowing the conditional distributions $f(X|Y)$ and $f(Y|X)$. The algorithm would proceed as follows.

`Specify an initial value` $X = X_0$

`Generate` Y_0 `from the conditional distribution` $f(Y|X = X_0)$

`For` $i = 1, \ldots, n$

`Generate` X_i `and` Y_i `from`

$$X_i \sim f(X|Y = Y_{i-1})$$
$$Y_i \sim f(Y|X = X_i)$$

One iteration of all the conditional distributions is called a sweep of the Gibbs sampler. As with most MCMC schemes a burn-in period is used. Thus, after the burn-in period has been implemented, the random variates that remain should be draws from the marginal distributions of X and Y. This scheme easily generalizes to the case of more than two variables by progressing in a systematic manner, taking each variable in turn and updating it according to its full conditional distribution. For a more in-depth explanation of the workings of the Gibbs sampler see Casella and George (1992).

The Gibbs sampling method introduced above is a systematic Gibbs sampler in that it systematically chooses the next random variate to be generated, i.e. always X and then Y. This can easily be adapted to become a random sweep

Gibbs sampler. This would mean that the order in which X_i and Y_i are updated is chosen at random. In the seminal paper by Geman and Geman (1984), a random sweep sampler was defined. For more details on random sweep samplers and methods of choosing the optimum selection probabilities see Levine and Casella (2006).

3.1.2 Metropolis-Hastings Algorithm

The Metropolis algorithm was introduced in Metropolis et al. (1953), where it was applied to the Boltzmann distribution. The method was then generalised in Hastings (1970) with respect to the proposal distributions used to produce candidate states for the chain to move to. Therefore, it became known as the Metropolis-Hastings algorithm. It was extensively used in physics for a long time before statisticians were alerted to its possibilities by Mueller (1991) and Tierney (1994). This algorithm is a very general method which is able to make draws from any probability distribution, assuming the target density can be calculated at a specific value. In simple terms, the method uses a proposal density, sometimes called the candidate generating density, to create a candidate state for the chain to move to. The chain then moves to this new state according to some probability, otherwise the chain stays at the current state. This method is easily shown in algorithmic form, but first some notation is required. Let X_i be the state of the chain at time i, X' be the candidate state, $\pi(X)$ be the target distribution of interest, and $Q(X, X')$ be the proposal density.

Specify an initial value X_0.

For $i = 0, 1, \ldots, n$.

Sample directly from $Q(X_i, X_i')$ a candidate state X_i'.

Generate a u from $Unif(0, 1)$.

Accept X_i' as X_{i+1} if $u \leq \alpha(X_i, X_i') = \min\left\{1, \frac{\pi(X_i')Q(X_i', X_i)}{\pi(X_i)Q(X_i, X_i')}\right\}$;

otherwise let $X_{i+1} = X_i$.

As is normal in MCMC, a burn-in period for the chain is required to enable the chain to reach its stationary distribution. A point to note about the Metropolis-Hastings transition kernel is that it is constructed in such a way that it is reversible. Some care has to be taken in defining the proposal density, so that the chain moves about the state space efficiently. A small summary of different types of proposal densities can be seen in Chib and Greenberg (1995). This paper is also relevant for a deeper explanation of the Metropolis-Hastings algorithm. An important point to note about the Metropolis-Hastings algorithm is that the proposal density $Q(X, X')$ satisfies detailed balance, see (3.2), by construction. This condition is sometimes referred to as the reversibility condition, and it requires the probability of a move from X_i to X_{i+1} to be the same as for the reverse move. Therefore, this is ensured in the algorithm with the introduction of $\alpha(X_i, X_i')$ to counteract any bias towards either state.

3.1.3 Reversible Jump MCMC

Following on from the Metropolis-Hastings and Gibbs schemes, there was a major development in MCMC research with the publication of Green (1995). This paper has somewhat revolutionised MCMC by defining an MCMC algorithm to cope with the situation, where the number of unknowns is itself also unknown. This new class of MCMC algorithms is commonly termed *reversible jump Markov chain Monte Carlo (RJMCMC)* but is sometimes referred to as *trans-dimensional MCMC*. It is essentially a Metropolis-Hastings algorithm for a much more general state space than standard Metropolis-Hastings. The state space is a union of subspaces, where the dimension of the subspaces may vary. It is easy to think of a state space like this arising in a problem of model selection, where a change in the model is a change in the subspace, and possibly a change of dimension. Most RJMCMC is implemented as one possible move in a hybrid sampler, where there are a number of different types of move allowed by the sampler at each iteration, so that the whole state space is sampled. Standard MCMC methods deal with the fixed-dimension sampling, but it is a RJMCMC move that allows the chain to jump between the different dimensions. Richardson and Green (1997) applied RJMCMC to finite mixture models with an unknown number of components. This is not the only variable dimensionality MCMC method. Stephens (2000a) defined a birth/death MCMC scheme as an alternative to RJMCMC also in a mixture context.

A quick summary of how the RJMCMC moves in Richardson and Green (1997) were developed is now given, because the 'new' moves to be defined later are similar in design. The model used by these authors differs significantly from

that given in section (2.2) by the fact that the component parameters and weight parameters have not been integrated out from the model. The authors proposed a move to split one component into two components, or conversely combine two into one, in the context of a mixture of normal distributions. A combine proposal was implemented by selecting two components j_1 and j_2 and bringing together their observations into a new component $j*$. The updating of the allocation variable was taken care of straightforwardly, but the weight parameter and component parameters for the new component had to be carefully constructed by moment matching. The reverse move of splitting $j*$ into j_1 and j_2 had to be designed so that the reversibility condition held for the split/combine move. Let y be the current state, let y' be the candidate state and let x be the data. The resulting acceptance probability for each of the moves has the form

$$\min\left\{1, \frac{\pi(y'|x)P(y' \to y)}{\pi(y|x)P(y \to y')}J(y, y')\right\}, \tag{3.3}$$

where $\pi(y|x)$ denotes the target density at state y and $P(y \to y')$ is the probability of proposing a move to y' from y. Finally, the term $J(y, y')$ is a Jacobian term which arises from the change in variable of going from y to y'.

3.2 Allocation Sampler

The sampler presented here runs across both k and g only, and it can therefore be thought of as a Reversible Jump sampler, since a change in k is essentially a change in the dimension of the model. However, the state space in this setting is all the possible allocation vectors, and thus, being a finite state space, is significantly simpler than that used in Richardson and Green (1997). The state spaces of the samplers reported in Richardson and Green (1997) and Stephens (2000a) contain all the parameters. Therefore, a change in k in their state spaces also means the number of parameters in the model changes which is not the case in our situation of having integrated the component parameters and weights from the model. Furthermore, the samplers defined in these two papers make use of both fixed k moves and variable k moves in order to try and move around the whole state space. This is the norm when developing an MCMC sampler for a complicated setting like this. Only using one type of move to sample from a distribution can lead to problems in being able to move around the whole state space from which one is trying to sample. Therefore, a hybrid sampler approach is used for the MCMC sampler in this thesis in order to try and maximise to efficiency. The sampler defined in this thesis will be referred to as the *allocation sampler*, because the allocation vectors play a very important role in the sampler as will become clear. The allocation sampler when approximating the posterior distribution (3.1) uses two types of move; moves that do not change the number of components, and moves that do. The sampler starts a move by firstly randomly selecting between these two types of move with equal probability. The first type of move consists of (i) Gibbs sampling on the components of g, (ii) three different

Metropolis-Hastings moves to simultaneously change several allocations and (iii) a Metropolis-Hastings move on the component labels. This helps the allocation sampler move more freely around the fixed k state space of allocation vectors. The allocation sampler that is used throughout assigns equal weighting to the two types of move.

3.2.1 Moves that do not change the number of components

3.2.1.1 Gibbs Move

This move is the implementation of a standard systematic sweep Gibbs sampling scheme that was introduced in Section 3.1.1. In this setting suppose the Markov chain at time t has k components and an allocation vector defined by $g = (g_1^{(t)}, \ldots, g_n^{(t)})$. A systematic sweep Gibbs sampler on the components of g, from g_1 to g_n proceeds as follows.

For $i = 1, \ldots, n$.

Compute $f_j(k, g = (g_1^{(t+1)}, \ldots, g_i = j, \ldots, g_n^{(t)}), x|\phi)$ for $j = 1, \ldots, k$.

Compute $p_j(g_i = j|k, g_{-i}, x, \phi) = \dfrac{f_j}{\sum_{j=1}^{k} f_j}$ for $j = 1, \ldots, k$,

where $g_{-i} = (g_1^{(t+1)}, \ldots, g_{i-1}^{(t+1)}, g_{i+1}^{(t)}, \ldots, g_{(n)}^{(t)})$

Sample $g_i^{(t+1)}$ from the discrete distribution $(p_1, \cdots, p_k)$

This Gibbs scheme move only changes one entry of g at a time, and thus one would expect strong serial dependence on the sampled g's, especially for moderate

to large sample sizes n. To combat this problem with the Gibbs sampling, moves in which more than one entry can be changed at once are proposed. The next three moves that are defined are formulated in this way through the Metropolis-Hastings algorithm.

3.2.1.2 Metropolis-Hastings Move 1

The first move that changes several allocations simultaneously is simple in its design. Essentially, all that the move is doing is taking the observations from two components and reallocating them to one of the two components with a given probability. To define the move in its algorithmic form, let g be the current state of the Markov chain and g' be the proposed candidate state. Then a move of this type proceeds as follows.

Randomly select 2 components j_1 and j_2 from the k available.
Make a draw p from the $Beta(\alpha_{j_1}, \alpha_{j_2})$ distribution.
For $i = 1, \ldots, n$
    If $g_i \in \{j_1, j_2\}$ then
        Allocate observation x_i to component j_1 with probability p
        or to component j_2 with probability $(1 - p)$.
Accept candidate allocation vector g' with probability equal
to $\min\{1, R\}$ where

$$R = \frac{f(k, g', x|\phi)}{f(k, g, x|\phi)} \frac{P(g' \to g)}{P(g \to g')}. \tag{3.4}$$

The Beta distribution $Beta(\alpha_{j_1}, \alpha_{j_2})$ is used to select the probability p, because

the parameters α_{j_1} and α_{j_2} are associated with the weight of the component. If there is no difference between the weights of the components, then the expectation of p is 0.5. Therefore, there is no bias towards any component, when the weights have a symmetric prior distribution, which is the normal setting used here.

The probability of moving from g to g', $P(g \to g')$ can be used to simplify R in the acceptance probability calculation. Since p is drawn from a $Beta(\alpha_{j_1}, \alpha_{j_2})$ distribution, integrating the following expression with respect to the distribution of p, we obtain

$$\begin{aligned} P(g \to g') &= \int \frac{\Gamma(\alpha_{j_1}+\alpha_{j_2})}{\Gamma(\alpha_{j_1})\Gamma(\alpha_{j_2})} p^{\alpha_{j_1}-1}(1-p)^{\alpha_{j_2}-1} p^{\tilde{n}_{j_1}}(1-p)^{\tilde{n}_{j_2}}\mathrm{d}p \\ &= \frac{\Gamma(\alpha_{j_1}+\alpha_{j_2})}{\Gamma(\alpha_{j_1})\Gamma(\alpha_{j_2})} \int p^{\alpha_{j_1}+\tilde{n}_{j_1}-1}(1-p)^{\alpha_{j_2}+\tilde{n}_{j_2}-1}\mathrm{d}p \\ &= \frac{\Gamma(\alpha_{j_1}+\alpha_{j_2})}{\Gamma(\alpha_{j_1})\Gamma(\alpha_{j_2})} \frac{\Gamma(\alpha_{j_1}+\tilde{n}_{j_1})\Gamma(\alpha_{j_2}+\tilde{n}_{j_2})}{\Gamma(\alpha_{j_1}+\alpha_{j_2}+\tilde{n}_{j_1}+\tilde{n}_{j_2})} \\ &= \frac{\Gamma(\alpha_{j_1}+\alpha_{j_2})}{\Gamma(\alpha_{j_1})\Gamma(\alpha_{j_2})} \frac{\Gamma(\alpha_{j_1}+\tilde{n}_{j_1})\Gamma(\alpha_{j_2}+\tilde{n}_{j_2})}{\Gamma(\alpha_{j_1}+\alpha_{j_2}+n_{j_1}+n_{j_2})}, \end{aligned} \tag{3.5}$$

where $\tilde{n}_{j_1}$, $\tilde{n}_{j_2}$ are the numbers of observations re-allocated to components j_1, and j_2. Now, comparing this to expression (2.10) for $f(g|k)$, it is evident that $P(g \to g')$ is essentially the contribution to $f(g'|k)$ for the two components j_1 and j_2. Therefore, the ratio of probabilities in (3.4) becomes

$$\frac{P(g' \to g)}{P(g \to g')} = \frac{f(g|k)}{f(g'|k)}. \tag{3.6}$$

Then, using (3.6), R in (3.4) simplifies to

$$R = \frac{f(x|k, g', \phi)}{f(x|k, g, \phi)}. \tag{3.7}$$

Consequently, as $f(x|k, g, \phi)$ is known at the current state, only $f(x|k, g', \phi)$ needs to be evaluated. In order to compute $f(x|k, g', \phi)$, only two terms in the product (2.15) have to be updated.

3.2.1.3 Metropolis-Hastings Move 2

The second of the Metropolis-Hastings moves proposed is again concerned with the observations of two randomly selected components. This move tries to move a group of observations from one component to another. This type of move makes sense, because, if the observations are already grouped into one component, then they may be similar in nature and thus it may be possible to the move a group of them as one to another component. The move proceeds in the following way.

Randomly select 2 components j_1 and j_2 from the k available.

If $n_{j_1} > 0$ then

Make a draw m from a discrete uniform distribution over the interval $[1, n_{j_1}]$.

Randomly select m observations from the n_{j_1} observations currently allocated to component j_1.

Move these m observations to component j_2 to produce a candidate allocation vector g'.

Accept g' with probability $\min\{1, R\}$ where

$$R = \frac{f(k, g', x|\phi)}{f(k, g, x|\phi)} \frac{P(g' \to g)}{P(g \to g')}. \tag{3.8}$$

In this case, the probability of proposing a transition from the current state g to

the candidate state g' is

$$P(g \to g') = \frac{1}{k}\frac{1}{k-1}\frac{1}{n_{j_1}}\binom{n_{j_1}}{m}^{-1} \tag{3.9}$$

$$= \frac{m!(n_{j_1}-m)!}{k(k-1)n_{j_1}n_{j_1}!}. \tag{3.10}$$

Furthermore, the probability of the associated reverse move is

$$P(g' \to g) = \frac{1}{k}\frac{1}{k-1}\frac{1}{n_{j_2}+m}\binom{n_{j_2}+m}{m}^{-1} \tag{3.11}$$

$$= \frac{m!n_{j_2}!}{k(k-1)(n_{j_2}+m)(n_{j_2}+m)!}. \tag{3.12}$$

Hence, the proposal ratio in (3.8) reduces to

$$\frac{P(g' \to g)}{P(g \to g')} = \frac{n_{j_1}}{n_{j_2}+m}\frac{n_{j_1}!n_{j_2}!}{(n_{j_1}-m)!(n_{j_2}+m)!}. \tag{3.13}$$

In the computation of the term $f(k, g', x|\phi)$ in (3.8), $f(x|k, g', \phi)$ and $f(g'|k)$ require to be calculated from (2.10) and (2.15). These calculations only require the terms to be changed which correspond to the two components taking part in the move, thus reducing the computational work.

3.2.1.4 Metropolis-Hastings Move 3

Finally, the third Metropolis-Hastings move is similar to the first move, in that two components are randomly selected, j_1 and j_2, and then the observations from these components are re-allocated to either of the components with a given probability. It is in the re-allocation probabilities that the two moves differ. The first move uses a constant probability across all the observations, but, in the third

move the probabilities of re-allocation change for each observation. Let $p_j^{(i)}, j \in \{j_1, j_2\}$ denote the probability of re-allocating the i-th observation to component j. The observations are processed for re-allocation in a random sequence, and the probabilities $p_j^{(i)}$ are calculated by defining them to be proportional to the probabilities that observation x_i is generated by component j, conditional on the value of x_i and on all previously newly re-allocated observations. To evaluate the probability of allocating an observation x_i to components j_1 or j_2, requires the following two conditions to be satisfied:

$$p_{j_1}^{(i)} + p_{j_2}^{(i)} = 1 \tag{3.14}$$

and

$$\begin{aligned} \frac{p_{j_1}^{(i)}}{p_{j_2}^{(i)}} &= \frac{f(g_i' = j_1 | \tilde{g}, x_i, \tilde{x}, k, \phi)}{f(g_i' = j_2 | \tilde{g}, x_i, \tilde{x}, k, \phi)} \\ &= \frac{f(g_i' = j_1, \tilde{g}, x_i, \tilde{x} | k, \phi)}{f(g_i' = j_2, \tilde{g}, x_i, \tilde{x} | k, \phi)} \\ &= \frac{f(g_i' = j_1, \tilde{g} | k)}{f(g_i' = j_2, \tilde{g} | k)} \frac{f(x_i, \tilde{x} | k, g_i' = j_1, \tilde{g}, \phi)}{f(x_i, \tilde{x} | k, g_i' = j_2, \tilde{g}, \phi)}, \end{aligned} \tag{3.15}$$

where $g' = (g_1', \ldots, g_n')$ is the candidate allocation vector being created by the move, $\tilde{x}$ is the vector of observations not in components j_1 and j_2, and all previously processed observations from components j_1 and j_2. Also, $\tilde{g}$ is the vector of allocations that corresponds to the observations contained in $\tilde{x}$. Now, using

(2.10) the first ratio from (3.15) is

$$\begin{aligned}
\frac{f(g_i' = j_1, \tilde{g}|k)}{f(g_i' = j_2, \tilde{g}|k)} &= \frac{\dfrac{\Gamma(\alpha_0)}{\Gamma(\alpha_0+n)} \displaystyle\prod_{j\in A^-} \frac{\Gamma(\alpha_j+n_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\alpha_{j_1}+\tilde{n}_{j_1}+1)\Gamma(\alpha_{j_2}+\tilde{n}_{j_2})}{\Gamma(\alpha_{j_1})\Gamma(\alpha_{j_2})}}{\dfrac{\Gamma(\alpha_0)}{\Gamma(\alpha_0+n)} \displaystyle\prod_{j\in A^-} \frac{\Gamma(\alpha_j+n_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\alpha_{j_1}+\tilde{n}_{j_1})\Gamma(\alpha_{j_2}+\tilde{n}_{j_2}+1)}{\Gamma(\alpha_{j_1})\Gamma(\alpha_{j_2})}} \\
&= \frac{\Gamma(\alpha_{j_1}+\tilde{n}_{j_1}+1)\Gamma(\alpha_{j_2}+\tilde{n}_{j_2})}{\Gamma(\alpha_{j_1}+\tilde{n}_{j_1})\Gamma(\alpha_{j_2}+\tilde{n}_{j_2}+1)} \\
&= \frac{(\alpha_{j_1}+\tilde{n_{j_1}})\Gamma(\alpha_{j_1}+\tilde{n_{j_1}})\Gamma(\alpha_{j_1}+\tilde{n_{j_2}})}{\Gamma(\alpha_{j_1}+\tilde{n_{j_1}})(\alpha_{j_2}+\tilde{n_{j_2}})\Gamma(\alpha_{j_1}+\tilde{n_{j_2}})} \\
&= \frac{\alpha_{j_1}+\tilde{n}_{j_1}}{\alpha_{j_2}+\tilde{n}_{j_2}}, \qquad (3.16)
\end{aligned}$$

where $A^- = \{j = 1, \ldots, k; j \neq j_1, j_2\}$. Also, the second ratio from (3.15) can be rewritten using (2.15) as

$$\frac{f(x_i, \tilde{x}|k, g_i' = j_1, \tilde{g}, \phi)}{f(x_i, \tilde{x}|k, g_i' = j_2, \tilde{g}, \phi)} = \frac{p_{j_1}(x_i, \tilde{x}^{j_1}|\phi_{j_1}) p_{j_2}(\tilde{x}^{j_2}|\phi_{j_2})}{p_{j_1}(\tilde{x}^{j_1}|\phi_{j_1}) p_{j_2}(x_i, \tilde{x}^{j_2}|\phi_{j_2})}. \qquad (3.17)$$

Now, substituting (3.16) and (3.17) into equation (3.15) one obtains

$$\frac{p_{j_1}^{(i)}}{1-p_{j_1}^{(i)}} = \frac{\alpha_{j_1}+\tilde{n}_{j_1}}{\alpha_{j_2}+\tilde{n}_{j_2}} \frac{p_{j_1}(x_i, \tilde{x}^{j_1}|\phi_{j_1}) p_{j_2}(\tilde{x}^{j_2}|\phi_{j_2})}{p_{j_1}(\tilde{x}^{j_1}|\phi_{j_1}) p_{j_2}(x_i, \tilde{x}^{j_2}|\phi_{j_2})}. \qquad (3.18)$$

Therefore, the allocation probabilities of an observation x_i are found by solving Equation (3.18) for $p_{j_1}^{(i)}$. After all the observations have been processed, a candidate allocation g' is accepted with the usual probability $\min\{1, R\}$, where

$$R = \frac{f(k, g', x|\phi)}{f(k, g, x|\phi)} \frac{P(g' \to g)}{P(g \to g')}. \qquad (3.19)$$

The ratio of the proposal probabilities in (3.19) can be written as

$$\begin{aligned}\frac{P(g' \to g)}{P(g \to g')} &= \frac{\frac{1}{k(k-1)} \prod_{i \in A} p_{g_i}^{(i)}}{\frac{1}{k(k-1)} \prod_{i \in A} p_{g'_i}^{(i)}} \\ &= \frac{\prod_{i \in A} p_{g_i}^{(i)}}{\prod_{i \in A} p_{g'_i}^{(i)}}, \end{aligned} \tag{3.20}$$

where $A = \{i : g_i = j_1 \cup g_i = j_2\}$. From (3.20) it is evident that the random order in which the observations are processed does not impact on the proposal ratio. The Metropolis-Hastings algorithm for this move can now be summarised as follows.

Randomly select two components j_1 and j_2 from the k available.

Randomly assign the observations from components j_1 and j_2 into a sequence $s = (s_1, \ldots, s_{n_{j_1}+n_{j_2}})$ in which they are to be processed.

For $m = 1, \ldots, (n_{j_1} + n_{j_2})$,

- Calculate $p_j^{(s_m)}, j \in \{j_1, j_2\}$ for observation s_m by solving (3.18).
- Make a draw of g'_{s_m}.
- Update quantities $\tilde{g}$, $\tilde{x}$, $n_{\tilde{j_1}}$ and $n_{\tilde{j_2}}$.

Accept the candidate allocation vector g' with probability $\min\{1, R\}$ where R is defined by (3.19).

3.2.1.5 Metropolis-Hastings labels move

This move essentially proposes to swap the labels of two components. It is an important move in the allocation sampler, because it allows the sampler to move more quickly between differing assignments of the labels. The Gibbs and Metropolis-Hastings moves defined in Sections 3.2.1.1 - 3.2.1.4 move relatively slowly between different assignments of the labels. Suppose the Markov chain is currently at the state $\{k, g\}$, with $\tilde{k}$ non-empty components in g. Then there are $\binom{k}{\tilde{k}}\tilde{k}!$ other possible allocation vectors that partition the data x in an identical way to g. Therefore, if the state $\{k, g\}$ has high posterior probability, then there are $k!/(k-\tilde{k}!)$ other allocation vectors that will also be similarly probable, and, in the case of a symmetric prior, will be equally probable *a posteriori*. In the asymmetric prior case, the current state of the chain may only be a local state of high posterior probability. A state with higher posterior probability may exist for the same partition of the data if the current assignment of the labels does not best match the prior for the groups in the data. Thus, a move that can speed up the movement of the sampler to these higher posterior probability states is required. A Metropolis-Hastings move is used to propose a change in the assignment of the labels of the allocation vector. The move proceeds in the following way.

`Randomly select two components` j_1 `and` j_2 `from the` k `available.`

`Generate a candidate allocation vector` g' `by swapping the labels of components` j_1 `and` j_2.

`Accept a move to the new labeling with probability` $\min\{1, R\}$

where

$$R = \frac{f(k, g', x|\phi)}{f(k, g, x|\phi)} \frac{P(g' \to g)}{P(g \to g')}. \tag{3.21}$$

The proposal ratio $P(g' \to g)/P(g \to g') = 1$ in (3.21) because the proposal probabilities are

$$P(g \to g') = \frac{1}{k} \frac{1}{k-1} = P(g' \to g).$$

In other words the proposal kernel is symmetric. Therefore, this means that (3.21) reduces to the ratio of the target density values at the two different labellings g and g',

$$R = \frac{f(k, g', x|\phi)}{f(k, g, x|\phi)}.$$

Furthermore, if this move was to be implemented in the symmetric case, it is obvious to see that $R = 1$ always. Therefore, because this move does not impact on the mixing on the chain in the symmetric case this move is only used in the asymmetric case. However, a problem of fixing a constant labeling structure to the components in the symmetric case arises, in order to carry out parametric inference, see Section 3.2.5.

3.2.2 Moves that change the number of components

This type of move, where there is a change in the number of components, can be thought of as a reversible jump move. In the finite mixture model (2.4), a change in the number of components corresponds to a change in the cardinality of the state space of the model. A model with k components has a cardinality of n^k elements. These moves have been designed using the Metropolis-Hastings algorithm in such a way that they make a pair of reversible moves. This pair of

moves is made up of an *ejection* and an *absorption* move. The ejection move is used to increase the number of components, and the absorption move to reduce the number of components. In other literature, the ejection/absorption terminology is replaced by split/combine. The ejection/absorption terminology is used here because it conveys better how the moves are constructed.

If the allocation sampler is implementing an ejection/absorption move, then the first task is to choose between an ejection move and an absorption move. The probability of an ejection move being proposed, when the current state of the chain is $\{k, g\}$, is

$$p_k^e = \begin{cases} 0 & k = k_{max} \\ \frac{1}{2} & k = 2, \ldots, (k_{max} - 1) \\ 1 & k = 1 \end{cases} \tag{3.22}$$

where k_{max} is the maximum number of components allowed in the model. This quantity should be chosen large enough so as not to stop the allocation sampler from moving to a number of components that possibly has a non-negligible probability. This bound is set to a default value of 50, but if the sampler nears this bound then it should be increased and the sampler re-run. If an ejection move is now attempted and a candidate state $\{k', g'\}$ is created, then the chain would move to this state with the usual Metropolis-Hastings acceptance probability form of $\min\{1, R\}$ where

$$R = \frac{f(k', g', x|\phi)}{f(k, g, x|\phi)} \frac{P(\{k', g'\} \to \{k, g\})}{P(\{k, g\} \to \{k', g'\})}. \tag{3.23}$$

In the reverse move, a candidate state is accepted as the next state of the chain according to the probability $\min\{1, R^{-1}\}$. The first point to notice for this acceptance probability of the reversible jump moves, is the lack of the Jacobian term that is usually a part of a RJMCMC scheme. A Jacobian term appears in the acceptance probability of a typical RJMCMC move because there is a change of variable taking place, which is usually characterised by a change in k changing the number of parameters in the model. However, in this case only the number of elements contained in the state space changes with a change in k, not the number of parameters, and therefore the Jacobian term is absent from (3.23).

These ejection/absorption moves have slightly different proposal schemes to create g', and thus proposal probabilities. The proposal schemes also depend on whether there is information distinguishing between the components, the asymmetric case, or there is no information, the symmetric case. As the asymmetric proposal is marginally easier to understand, this will be defined first.

3.2.2.1 Asymmetric case

Let the current state of the sampler be $\{k, g\}$. Then an ejection move that increases the number of components in the model by 1 proposes to move to a candidate state $\{k', g'\}$. The move proceeds as follows.

Randomly select an ‘‘ejecting component’’, j_1, from k available.
Create a new component, j_2, with label $k+1$.
Draw a probability p_E from a $Beta(a, a)$ distribution.
With probability $(1 - p_E)$ re-allocate each of the n_{j_1} observations in component j_1 to component j_2 , otherwise leave in j_1.

Accept the candidate state $\{k', g'\}$ with probability $\min\{1, R\}$ where R is defined by (3.23).

The justification for using a draw from a $Beta(a, a)$ distribution as the probability of ejection p_E and how to choose a are discussed in Section 3.2.3.

Let $\tilde{n}_{j_1}$ and $\tilde{n}_{j_2}$ be the numbers of observations allocated to the ejecting and the ejected components. Then the probability $P(\{k, g\} \to \{k', g'\})$ of moving from the current state to the candidate state, after integrating with respect to the distribution of p_E, is formulated as follows:

$$\begin{aligned} P(\{k, g\} \to \{k', g'\}) &= \int p_k^e \frac{1}{k} \frac{\Gamma(a+a)}{\Gamma(a)\Gamma(a)} p_E^{a-1}(1-p_E)^{a-1} p_E^{\tilde{n}_{j_1}}(1-p_E)^{\tilde{n}_{j_2}} \mathrm{d}p_E \\ &= p_k^e \frac{1}{k} \frac{\Gamma(2a)}{\Gamma(a)\Gamma(a)} \int p_E^{a+\tilde{n}_{j_1}-1}(1-p_E)^{a+\tilde{n}_{j_2}-1} \mathrm{d}p_E \\ &= p_k^e \frac{1}{k} \frac{\Gamma(2a)}{\Gamma(a)\Gamma(a)} \frac{\Gamma(a+\tilde{n}_{j_1}) + \Gamma(a+\tilde{n}_{j_2})}{\Gamma(2a+n_{j_1})}. \end{aligned} \tag{3.24}$$

The proposal of the reverse absorption move is very simple in the asymmetric case. Starting from the state $\{k' = k+1, g'\}$, the candidate state is generated using the following scheme.

Set the ‘‘absorbed component’’, $j_2 = k' = k+1$

Randomly select the ‘‘absorbing component’’ j_1 from the k available components.

Move all the n_{j_2} observations from component j_2 to component j_1 to create a candidate state.

Accept the candidate state with probability $\min\{1, R^{-1}\}$ where R

`is defined by` (3.23).

The probability of proposing this move $P(\{k', g'\} \to \{k, g\})$ is clearly,

$$P(\{k', g'\} \to \{k, g\}) = (1 - p_k^e)\frac{1}{k}. \tag{3.25}$$

Therefore, the ratio of the proposal probabilities in (3.23) is

$$\frac{P(\{k', g'\} \to \{k, g\})}{P(\{k, g\} \to \{k', g'\})} = \frac{1 - p_k^e}{p_k^e} \frac{\Gamma(a)\Gamma(a)}{\Gamma(2a)} \frac{\Gamma(2a + n_{j_1})}{\Gamma(a + \tilde{n}_{j_1})\Gamma(a + \tilde{n}_{j_2})}. \tag{3.26}$$

The computation of $f(k', g', x|\phi)$ in (3.23) requires only changing two terms in (2.10) and (2.15) in a similar fashion to the second fixed-k Metropolis-Hastings move.

3.2.2.2 Symmetric case

As stated above, the proposal scheme for the symmetric case differs slightly from the asymmetric case. The symmetric case has the added feature that, in the absorbing move, both the absorbing and absorbed components are selected randomly. For detailed balance to still hold, a change is also required to the ejection procedure. To keep the chain reversible, the ejection move is required to allow the ejected component to be any of the $k + 1$ components available and is therefore not forced to be the $(k + 1)^{th}$ component as in the symmetric case. This can be achieved by including in the ejection move a swap between the label $j_2 = k+1$ of the ejected component and the label of a randomly selected component, including the ejected component itself. This increased random aspect in the symmetric

case proposal is implemented to enhance the mixing ability of the sampler. It impacts on the proposal probabilities (3.24) and (3.25) by multiplying both of these terms by $(k+1)$ and therefore the ratio of these probabilities stays the same as (3.26).

Some remarks are required on the actual implementation of this scheme for the symmetric case. It is a possibility that, as a consequence of an absorption move, a gap in the sequence of components arises. This problem could be easily solved by either changing the label of the highest labelled component to that of the missing label in the sequence, or by decreasing the labels by one of all the components greater than the absorbed component. These methods would not be removing an empty component from the model but only swapping the labels. However, these solutions are not needed because in the symmetric case, where all hyperparameters are equal, the labels of the components are just placeholders and have no influence on the workings of the sampler. Computation time can therefore be saved by thus allowing gaps to appear in the sequence of components, but a vector storing the components which are in the mixture at stage of the sampler is required.

3.2.3 Ejecting Probability, p_E

The selection of the ejecting probability, p_E, obviously has a bearing on the mixing properties of the sampler. This probability, as seen in the above moves, is selected using a random draw from a $Beta(a, a)$ distribution. This procedure was decided upon after some experimentation. Two of the other methods tested were (i) a constant value of $p_E = 0.5$, and (ii) a random draw from a $Unif(0, 1)$ distribution.

To illustrate why the $Beta(a, a)$ method was chosen, a simple comparison of the three methods was carried out. The allocation sampler was run using three different datasets which highlighted problems emanating from the selection of p_E. The sampler was run implementing each different method of selecting p_E for each dataset using no burn-in and 500000 moves with a thinning parameter, $\Delta = 10$, thus creating a sample of 50000. The three datasets used were (a) the galaxy dataset, see Section (5.1) for more details on this dataset, (b) a random sample of 2000 from the claw distribution described in Table (4.1) and (c) a random sample of 200 from a six-component equally weighted mixture of 10-dimensional multivariate normals. For mixture density (c) the components' means were $m_{1i} = m_{2i} = m_{6i} = 0$, $m_{3i} = 2$, $m_{4i} = -2$, $m_{5i} = (-1)^i$, $i = 1, \ldots, 10$. As for the covariance matrices, we used $r_1^{-1} = r_3^{-1} = r_4^{-1} = r_5^{-1} = I$, $r_2^{-1} = 0.25I$ and $(r_6^{-1})_{ij} = 0.9^{|i-j|}$, $i, j = 1, \ldots, 10$, where I denotes the 10-dimensional identity matrix. Figures (3.1) - (3.3) show trace plots of k for the sample of 50000 for each dataset with the three different ways of selecting p_E. For the galaxy dataset, Figure(3.1) shows that all the methods do enable the Markov chain to make jumps between dimensions, but Table 3.1 shows that the mixing of the chain is better in the $Beta(a, a)$ case because the effective sample size is considerably higher. In the more complicated example of Figure (3.2), the $Unif(0, 1)$ method finds it extremely hard to move away from the starting position of 1 component. The constant p_E and $Beta(a, a)$ methods both manage to reach equilibrium rather quickly, but again the $Beta(a, a)$ seems to mix better after that point. Finally, Figure (3.3) highlights a problem situation for the constant p_E. This time the $Unif(0, 1)$ and $Beta(a, a)$ methods perform reasonably well compared with the constant p_E, being unable to move from the 1 component model for the sample

of 50000. Therefore, from these examples the method of drawing p_E from a $Beta(a, a)$ distribution is preferable, as it seems to reach equilibrium whatever the structure of the data and, also, the Markov chain seems to mix to better, using this method.

Dataset	$p_E \sim Beta(a, a)$	$p_E \sim Unif(0, 1)$	$p_E = \frac{1}{2}$
Galaxy	1268	816	734
Claw	755	-	73
Six MVN	1153	483	-

Table 3.1: Comparison of effective sample size once equilibrium is reached for 3 different p_E probability selection methods across 3 datasets. Note that equilibrium was not reached in two cases. See Appendix B for details of how the effective sample size is calculated.

Some further remarks about the $Beta(a, a)$ distribution are required. The selection of the parameter a can have a significant effect on the performance of the sampler. It was chosen to ensure that empty components were proposed relatively often. This helps the allocation sampler to jump between models with different numbers of components more easily. Therefore, with $p_E \sim Beta(a, a)$, a is selected according to $Pr[\tilde{n}_{j_2}] = \frac{p_0}{2}$, where p_0 is the probability of either ejecting all the observations from the ejecting component, j_1, into the new component j_2, or ejecting no observations into the new component. Recall that $\tilde{n}_{j_2}$ is the number of observations that are re-allocated to component j_2. After some experimentation into a suitable value for p_0, $p_0 = 0.2$ was selected because it gave the most satisfactory results. This means that the probability that the new ejected component j_2 is empty with probaility 0.1. Therefore, the allocation sampler allows empty components to occur in the model. So, to find a the following equation

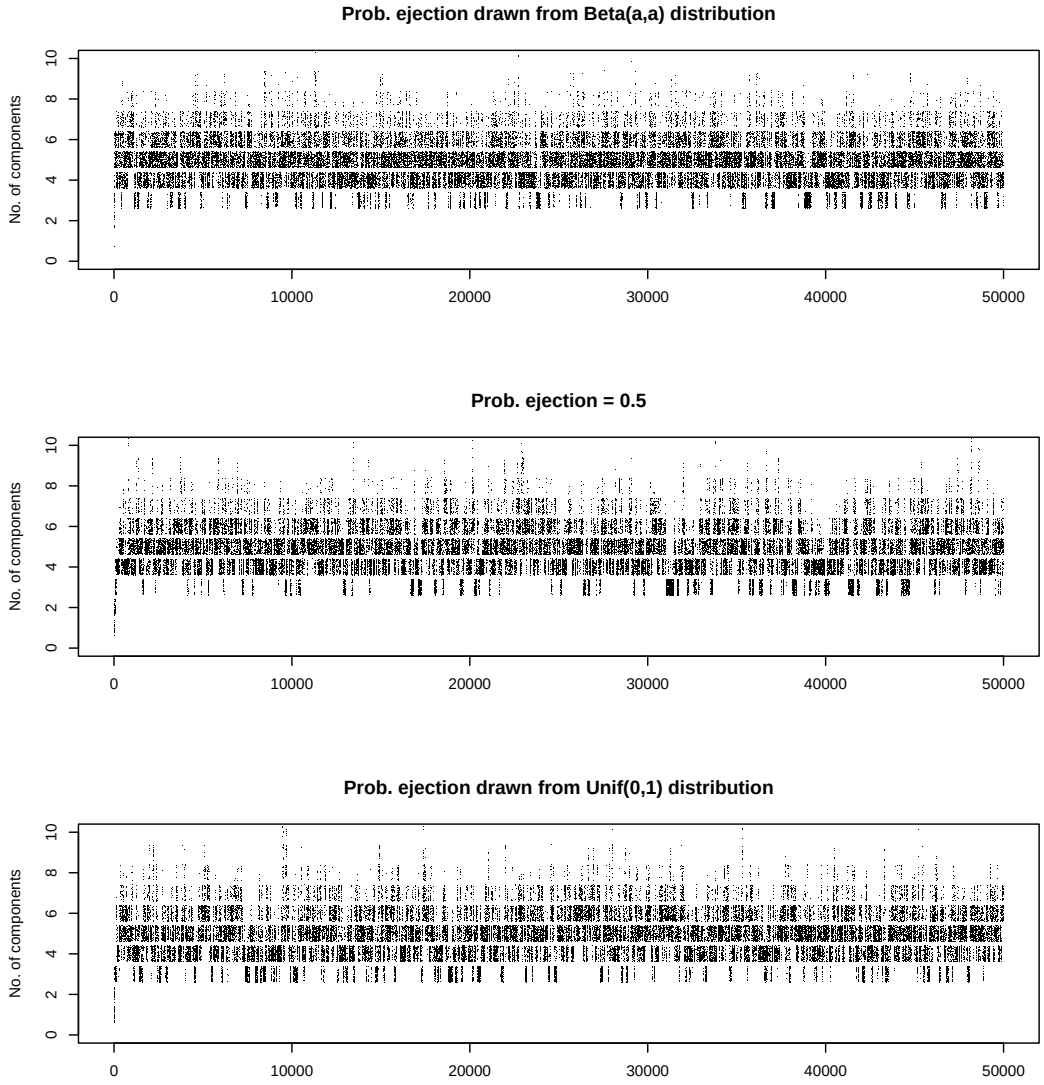

Figure 3.1: Trace plots of k for the galaxy dataset corresponding to 3 different ways of selecting the probability of ejection p_E values. The value of a in the top graph is not fixed and changes throughout the simulation according to Equation (3.28).

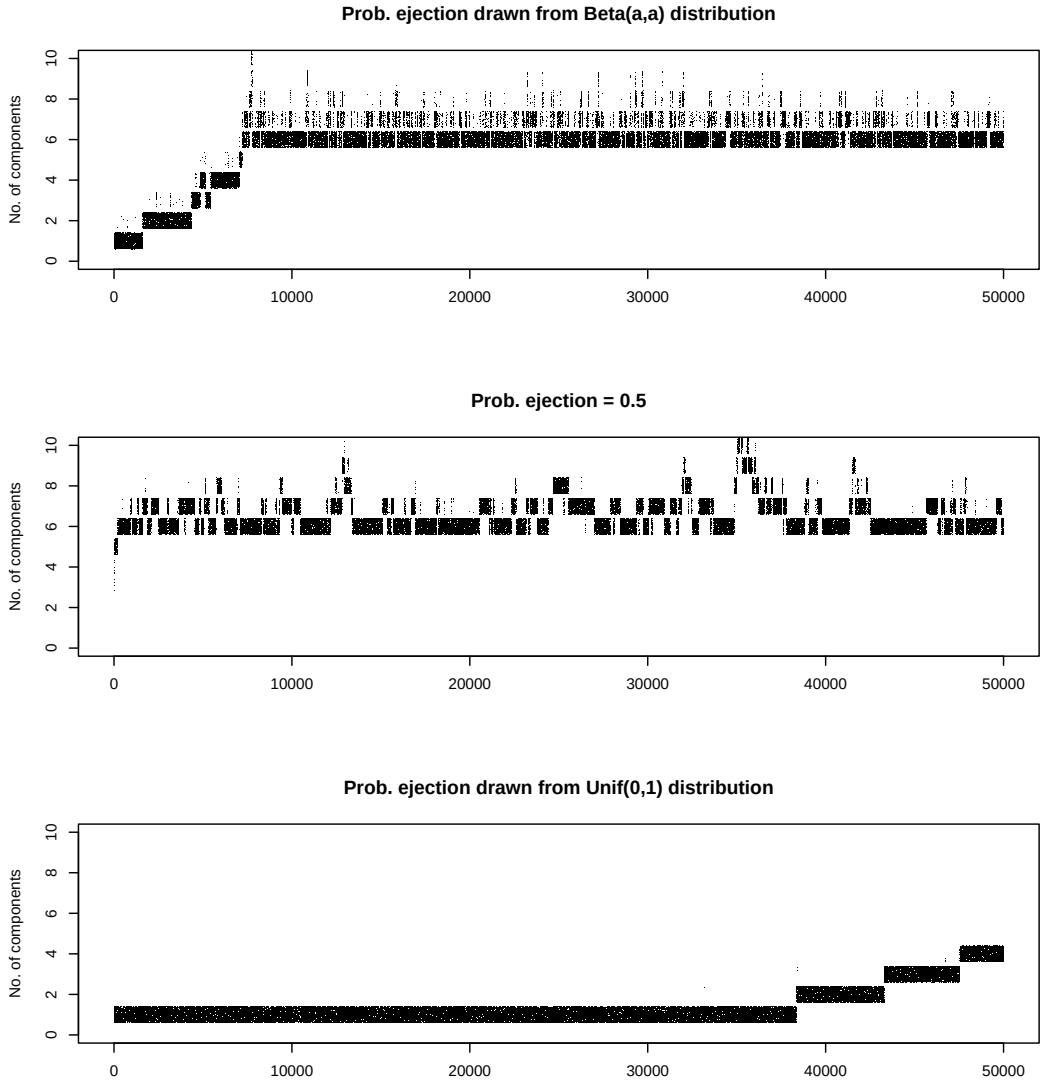

Figure 3.2: Trace plots of k for the claw dataset corresponding to 3 different ways of selecting the probability of ejection p_E values. The value of a in the top graph is not fixed and changes throughout the simulation according to Equation (3.28).

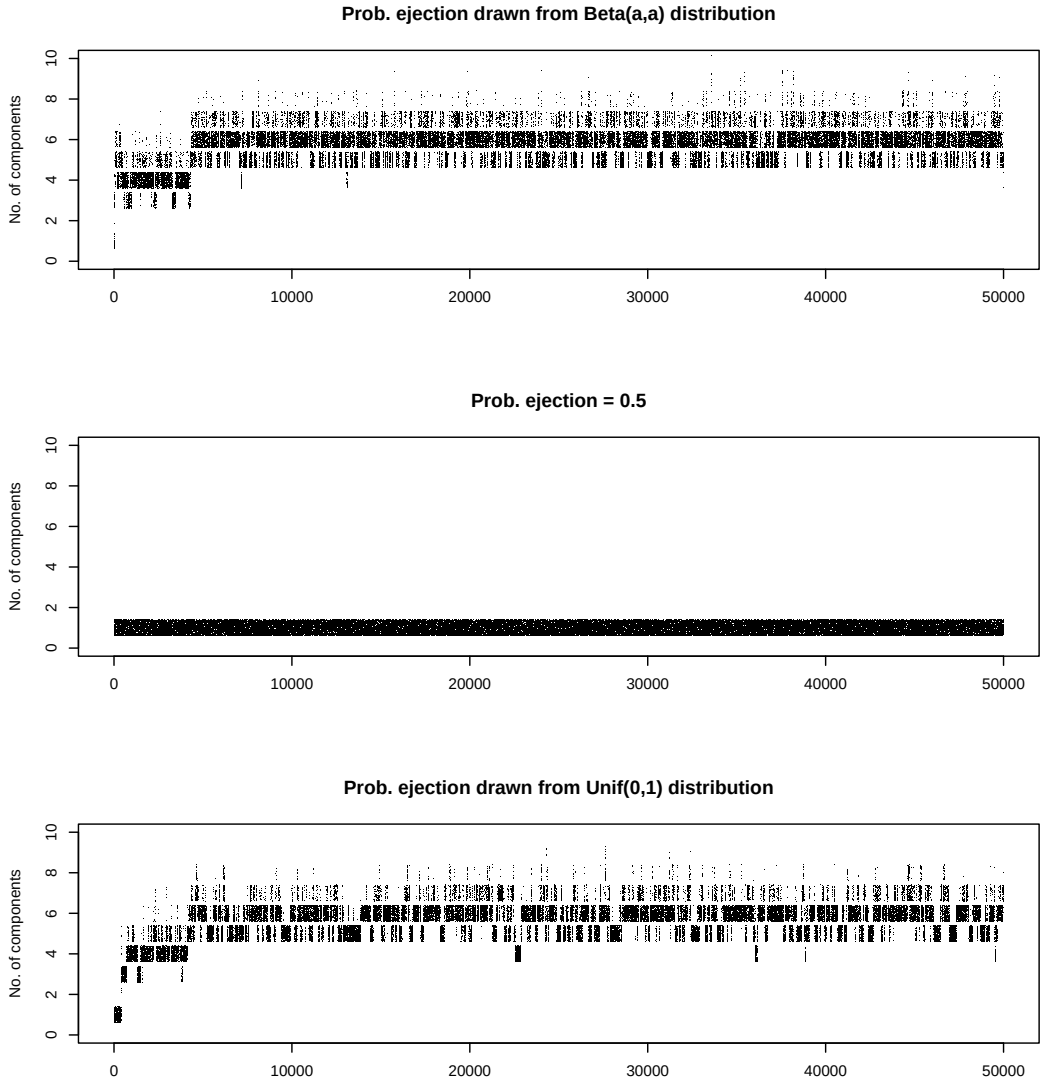

Figure 3.3: Trace plots of k for the six component 10-dimensional multivariate normal dataset corresponding to 3 different ways of selecting the probability of ejection p_E values. The value of a in the top graph is not fixed and changes throughout the simulation according to Equation (3.28).

has to be solved for a:

$$\begin{aligned}
\frac{p_0}{2} &= \int_0^1 (1-p_E)^{n_{j_1}} \frac{\Gamma(2a)}{\Gamma(a)\Gamma(a)} p_E^{a-1}(1-p_E)^{a-1}\,\mathrm{d}p_E \\
&= \frac{\Gamma(2a)}{\Gamma(a)\Gamma(a)} \int_0^1 (1-p_E)^{n_{j_1}+a-1} p_E^{a-1}\,\mathrm{d}p_E \\
&= \frac{\Gamma(2a)}{\Gamma(a)\Gamma(a)} \frac{\Gamma(a)\Gamma(a+n_{j_1})}{\Gamma(2a+n_{j_1})} \\
&= \frac{\Gamma(2a)}{\Gamma(a)} \frac{\Gamma(a+n_{j_1})}{\Gamma(2a+n_{j_1})}. \qquad (3.27)
\end{aligned}$$

The right-hand side of Equation (3.27) can be re-written as

$$\begin{aligned}
\frac{\Gamma(2a)}{\Gamma(a)} \frac{\Gamma(a+n_{j_1})}{\Gamma(2a+n_{j_1})} &= \frac{\Gamma(2a)}{\Gamma(a)} \frac{(a+n_{j_1}-1)(a+n_{j_1}-2)\ldots(a)\Gamma(a)}{(2a+n_{j_1}-1)(2a+n_{j_1}-2)\ldots(2a)\Gamma(2a)} \\
&= \frac{1}{2}\left(\frac{a+n_{j_1}-1}{2a+n_{j_1}-1}\right)\left(\frac{a+n_{j_1}-2}{2a+n_{j_1}-2}\right)\cdots\left(\frac{a+1}{2a+1}\right). \qquad (3.28)
\end{aligned}$$

It is evident from Equation (3.28) that the value of a will change throughtout the simulation because it is dependent on n_{j_1}. Furthermore, if the majority of values of a are close to 1, which is the case in Figure 3.1, then the $Beta(a,a)$ method will perform similarly to the $Unif(0,1)$ method because the $Unif(0,1)$ distribution corresponds to a $Beta(1,1)$ distribution. Equation (3.28) is a product of $(n_{j_1}-1)$ terms that have the form $\frac{a+n_{j_1}-c}{2a+n_{j_1}-c}$, where $c=1,\ldots,(n_{j_1}-1)$. These terms can be shown to be monotonically decreasing in a. It is simply shown by differentiating each term with respect to a and noticing that each derivative is less than zero. Therefore, this means that Equation (3.27) can be solved relatively easily by a numerical method such as the bisection algorithm. However, this can

be somewhat time-consuming, so, to reduce the calculation time, Equation (3.27) was solved only for n_{j_1} over a grid of values, equally spaced on a log-scale. These solutions were then stored inside the simulation program. Hence, whenever a p_E probability is required, the simulation program finds the appropriate value of $\log a$ for the current value of $\log n_{j_1}$, using linear interpolation of the solutions for the nearest n_{j_1}'s in the grid.

3.2.4 Hyperparameters

Here the problem of selecting the component hyperparameter values for the sampler is addressed. When there is some information about the components this should be incorporated into the prior distributions. The hyperparameters should then reflect this knowledge about the components. This would be an instance of the asymmetric setting. However, in the symmetric case it can be more difficult to set hyperparameter values, and therefore a general method is required for this setting. The selection of these hyperparameters ϕ plays an important role in the implementation of the allocation sampler because the hyperparameters can have a significant effect on the posterior distributions. Richardson and Green (1997), Nobile (2005) and Jasra et al. (2005) all illustrate the effect of the hyperparameters on the posterior distribution of k for the galaxy dataset when using normal components, through a sensitivity analysis. All of their analyses show how the posterior distribution of k can change considerably with a change in the hyperparameter values. In Richardson and Green (1997) the authors chose to impose a hyperprior on the hyperparameter β in their univariate normal mixture model in order to try and reduce the effects of the hyperparameter values on the posterior

distributions. The Bayesian finite mixture model used in Nobile (2005) is the same as that used throughout this thesis and his approach is adopted and extended in this thesis. Nobile (2005) commented that the main reason for using his method is that the marginal posterior distributions of the hyperparameters have very long tails and therefore the approach of using random hyperparameter values throughout the simulation has a large impact on the posterior distributions.

Every family of components has a slightly different method due to the different hyperparameters in the model. A method using a preliminary run of the sampler, where some or all of the hyperparameters have a hyperprior placed on them, is used in this thesis. The output from this preliminary run is then used to fix the hyperparameters for subsequent runs. In the preliminary run of the allocation sampler samples from the posterior distributions of certain hyperparameters are produced using Metropolis-Hastings moves. A move of this type is proposed at every iteration of the allocation sampler. The hyperparameters are then fixed, by looking at the sample made from the distribution of hyperparameters in the preliminary run. The hyperpriors used for each of the different families of components are as follows.

3.2.4.1 Mixtures of univariate normals

In the case of univariate normal components, where the component hyperparameters are $\phi = (\mu, \tau, \gamma, \delta)$, hyperpriors are only placed upon two of these hyperparameters, namely τ and δ. The other two hyperparameters, μ and γ, are fixed throughout the sampler. The sample mean $\bar{x}$ is taken for μ and γ is set equal to 2. This value of γ has been chosen because the prior predictive distribution is

a t distribution with 2γ degrees of freedom, and thus setting $\gamma = 2$ means that the prior predictive has 4 degrees of freedom leading to relatively thick tails, but finite second moments. Independent hyperpriors are imposed on τ and δ and have the form

$$(1+\tau)^{-1} \sim Unif(0,1)$$
$$\delta \sim Unif(0, \delta_U),$$

where $\delta_U = (\gamma - 1)s_x^2$ and s_x^2 is the sample variance. The starting values used for τ and δ in the preliminary run are $\tau = 0.5$ and $\delta = 0.5s_x^2$.

3.2.4.2 Mixtures of multivariate normals

A similar method is applied for the hyperparameters of multivariate normal components, $\phi = (\mu, \tau, \nu, \xi)$. Again, μ is fixed and is simply set as the sample mean vector. The prior predictive distribution in this case is a multivariate t distribution with $\nu - b + 3$ degrees of freedom, which when set equal to 4, so that it corresponds to the degrees of freedom in the univariate case, yields $\nu = b + 3$, where b is dimension of the data. The other two hyperparameters τ and ξ have independent hyperpriors placed upon them in a similar way to the univariate case. Firstly, τ uses the same $(1+\tau)^{-1} \sim Unif(0,1)$ form. However, one assumes that ξ is a diagonal matrix and each diagonal entry is dealt with in the same way as δ in the univariate case.

3.2.4.3 Mixtures of uniforms

For the setting of uniform components where there exists only the one hyperparameter ϕ, a hyperprior of the form

$$\frac{1}{\phi} \sim Unif(0, u)$$

is implemented, where $u = \frac{1}{\max_i |x_i|}$.

3.2.4.4 Mixtures of sign-shifted exponentials

Finally, for sign-shifted exponential components three out of the four hyperparameters use the hyperprior approach in the preliminary run. The independent hyperpriors used for these hyperparameters are

$$\begin{aligned} \log(\beta) &\sim Unif(-\bar{\beta}, \bar{\beta}) \\ \log(\gamma) &\sim Unif(-\bar{\gamma}, \bar{\gamma}) \\ \rho &\sim Unif(0, 1), \end{aligned} \tag{3.29}$$

where $\bar{\beta}$ and $\bar{\gamma}$ are chosen to be reasonably large values; in most cases $\bar{\beta} = \bar{\gamma} = 50$ is sufficient. The choice of the hyperprior on ρ makes sense because ρ is a probability. In addition, the other hyperparameter κ is fixed for all runs at the value $\kappa = 1$, which means that the prior on ω reduces to an exponential distribution.

A graphical method is used to select the single hyperparameter value which will be used in subsequent runs of the sampler. Firstly, a trace plot of k is inspected and all the draws from the distributions of hyperparameters before equilibrium is reached are removed for subsequent calculations. This is essentially adjusting the burn-in period of the Markov chain. The hyperparameter value is then chosen by producing boxplots of the marginal posterior distributions of the draws of hyperparameter values from the preliminary run of the sampler conditional on k. Estimates of the hyperparameters are then computed using the medians of the posterior draws, utilizing only the draws that correspond to values of k after a rough leveling off of the medians has occurred.

3.2.4.5 Preliminary run settings

The preliminary run settings used throughout produce a sample of 50000 draws using a burn-in of 10000, and then a further 500000 draws are made with a thinning parameter Δ equal to 10. The idea of thinning the draws is to improve the mixing of the Markov chain. Thus, in the preliminary run every tenth draw is saved with the rest being discarded, to end up with a sample of size 50000. Any preliminary run is carried out with these standard settings to produce samples of size 50000 for respective hyperparameters.

To illustrate how the above procedure is implemented an example using the galaxy dataset, see Section 5.1 for data description, is shown in Figures 3.4 and 3.5. Figure 3.4 provides evidence that there is correlation between the hyperparameters and the number of components in the model. Looking closely, it can be seen that, for τ, the values increase for smaller values of k. This can be explained

by the variance of the data being contained within the components, rather than between the components for small values of k. Figure 3.5 is used to specify the τ and δ values for subsequent runs of the sampler. In the top row one can see that the medians of the boxplots level off with the number of components greater than or equal to 3 giving an estimate for τ equal to 0.04. Also, in the bottom row there is a rough leveling off after 6 components, which corresponds to an estimate for δ equal to 2.

Also, the thinning parameter Δ for the subsequent runs is chosen using the output from the preliminary run. Its value is selected by aiming for a lag 1 autocorrelation of 0.7 in the sampled values of k. This thinning value would then produce a lag-1 autocorrelation of similar size in the actual runs.

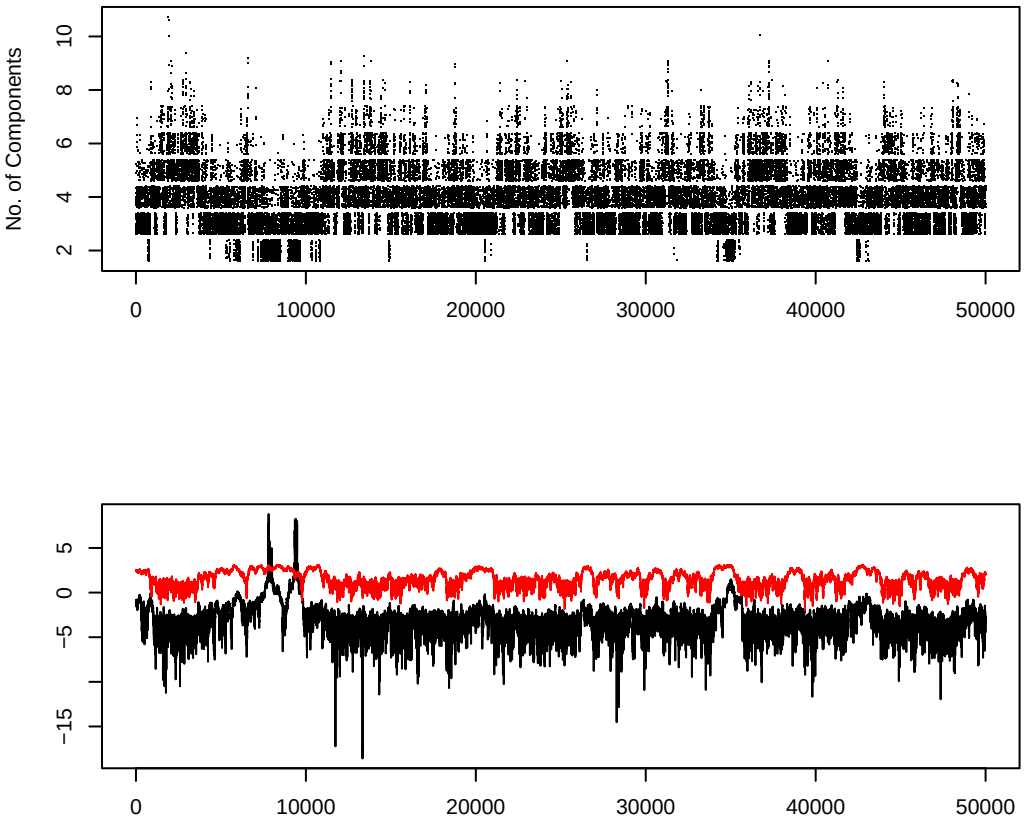

Figure 3.4: The top graph shows a trace of k for a preliminary run of the allocation sampler using the galaxy dataset, and the lower graph displays a trace of the hyperparameters τ and δ for the preliminary run. The red line corresponds to the trace for δ and the black line for τ.

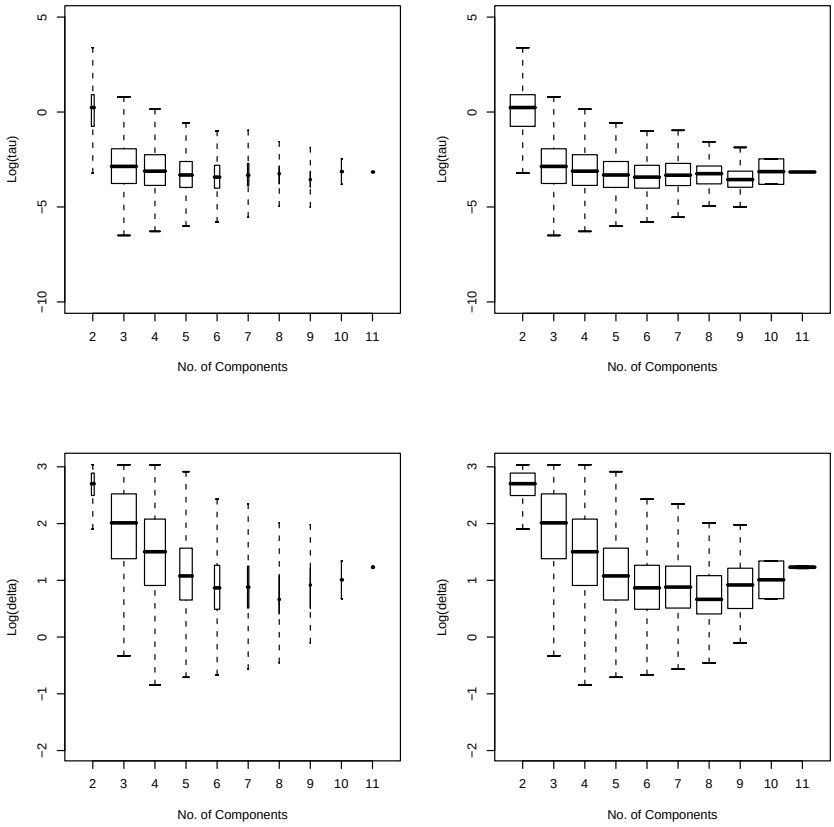

Figure 3.5: The top row displays boxplots of draws from the posterior distribution, conditional on k, for τ where the left graph has the boxplot size weighted according to the frequency of k. The bottom row shows the same distributions for δ.

3.2.4.6 Metropolis-Hastings Hyperparameter moves

The construction of these moves encounters some practical implementation problems that have to be addressed when calculating the Metropolis-Hastings acceptance probability. It should be noted that in some of the priors detailed in Section 3.2.4 the prior is placed on a function of the hyperparameter and not on the hyperparameter itself. It should be noted that these changes of variables do not impact on the acceptance probability. If more than one hyperparameter is being proposed to change, then the proposals are made independently. A move of this type progresses in the following way.

`Propose a new hyperparameter value for all required hyperparameters to create a candidate state` ϕ_2.

`Accept the move from the current state,` ϕ_1, `to the candidate state with probability` $\min\{1, R\}$ `where`

$$\begin{aligned} R &= \min\left\{1, \frac{f(x,k,g,\phi_2)}{f(x,k,g,\phi_1)}\frac{P(\phi_2 \to \phi_1)}{P(\phi_1 \to \phi_2)}\right\} \\ &= \min\left\{1, \frac{f(x|k,g,\phi_2)f(k,g)\pi(\phi_2)}{f(x|k,g,\phi_1)f(k,g)\pi(\phi_1)}\frac{P(\phi_2 \to \phi_1)}{P(\phi_1 \to \phi_2)}\right\} \\ &= \min\left\{1, \frac{f(x|k,g,\phi_2)}{f(x|k,g,\phi_1)}\frac{P(\phi_2 \to \phi_1)}{P(\phi_1 \to \phi_2)}\right\}. \end{aligned} \tag{3.30}$$

A practical problem arises with the proposal distribution and hence the proposal probabilities $P(\phi_2 \to \phi_1)$ and $P(\phi_1 \to \phi_2)$. Suppose a proposal of a candidate state is being made for an arbitrary hyperparameter ϖ, that takes values on an interval H. Then, as stated above, it is done using a random draw from

a uniform distribution. The simplest case would use a distribution centred on the current value ϖ_1, i.e. $Unif(\varpi - \varepsilon, \varpi + \varepsilon)$, where ε is chosen to be 1% of the range of H. Then, since this proposal distribution is symmetric, the ratio of proposal probabilities in (3.30) would simply equal 1. However, occasionally a move will be rejected straight away due to the candidate state ϖ_2 lying outside H. To overcome this problem of proposing unsuitable states, and to improve the mixing of the Markov chain, a different distribution is used when the current or candidate or both states are close to an endpoint of H. The proposal distribution used in these cases can be summarised by a mixture of two uniform distributions. Furthermore, the proposal distribution, used in any case, for a move from ϖ_1 to ϖ_2 can be defined as

$$\frac{1}{2}Unif(\max(H_L, \varpi_1 - \varepsilon), \varpi_1) + \frac{1}{2}Unif(\varpi_1, \min(H_U, \varpi_1 + \varepsilon)), \tag{3.31}$$

where H_L and H_U are the lower and upper limits of the range space H. This modification to the simple proposal case is implemented because if ϖ gets close to an endpoint it may get stuck there and find it hard to leave that vicinity.

The ratio of proposal probabilities in (3.30) is simply a product of the proposal probability ratios for each hyperparameter. These ratios are dependent on whether the proposed candidate state is in a positive or negative direction. The direction of proposal is chosen with probability equal to $\frac{1}{2}$. For a move to a candidate state in a positive direction the proposal probability is

$$P(\varpi_1 \to \varpi_2) = \frac{1}{2}\frac{1}{\min(H_U, \varpi_1 + \varepsilon) - \varpi_1}. \tag{3.32}$$

Therefore, the proposal probability of the corresponding reverse move from ϖ_2 to ϖ_1 is

$$P(\varpi_2 \to \varpi_1) = \frac{1}{2}\frac{1}{\varpi_2 - \max(H_L, \varpi_2 - \varepsilon)}. \tag{3.33}$$

Thus, the proposal probability ratio for this type of move can be written as

$$\frac{P(\varpi_2 \to \varpi_1)}{P(\varpi_1 \to \varpi_2)} = \frac{\min(H_U, \varpi_1 + \varepsilon) - \varpi_1}{\varpi_2 - \max(H_L, \varpi_2 - \varepsilon)}. \tag{3.34}$$

Also, if the proposed candidate state is made in the negative direction then the associated ratio of proposal probabilities is

$$\frac{P(\varpi_2 \to \varpi_1)}{P(\varpi_1 \to \varpi_2)} = \frac{\varpi_1 - \max(H_L, \varpi_1 - \varepsilon)}{\min(H_U, \varpi_2 + \varepsilon) - \varpi_2}. \tag{3.35}$$

3.2.5 Label switching problem

Finite mixture distributions are not identifiable because the likelihood function for a mixture model is invariant to a permutation of the labels of the components in the model:

$$L(\lambda, \theta; x) = \prod_{i=1}^{n} \{\lambda_1 q(x_i|\theta_1) + \cdots + \lambda_k q(x_i|\theta_k)\} \tag{3.36}$$

For example, in a mixture of two components, whether the components are labeled $\{1, 2\}$ or $\{2, 1\}$ has no bearing on the likelihood value defined by (3.36). This is what was defined as the *label-switching* problem in Redner and Walker (1984). This lack of identifiability creates no problem for predictive inference since the components' labelling has no bearing on a predictive density, see (2.48) and (2.49) for examples. However, if parameter estimation or classification is of interest, then this problem has to be addressed. A solution to this problem requires the mixture components to have an unequivocal assignment of the labels in order to remove the highly symmetrical form of the posterior distributions. If the prior distribution used is symmetric, then the resulting posterior distributions will also have this complete symmetric feature. In the asymmetric case where there is information distinguishing the components in the prior distribution, this problem is taken care of by a Metropolis-Hastings move on the labels, see Section 3.2.1.5 for further details.

An example illustrating the symmetry in the posterior distributions can be seen in Figure 3.6. This figure displays the marginal posterior distributions of the means for the galaxy dataset, see Section 5.1 for data description, conditional

on there being 3 univariate normal components. It is evident that all the distributions in the top row of plots are very similar. Each graph has three peaks that correspond to the means of the three components in this model. This occurs because during the running of the MCMC sampler the components are frequently swapping labels. The top row of Figure (3.6) shows the effects of not tackling the label-switching problem on the posterior distributions. The bottom row of the figure shows how each component can be extracted if the label-switching problem is addressed. See Section 3.2.5.1 for details of procedure used here.

There have been a number of different methods proposed to counteract the problem of label switching. A common approach to the problem is to impose identifiability constraints on the parameter space. These constraints could be imposed on the component weights/means/variances, or a combination of these parameters. For example, the component means could be ordered using the constraint, $\mu_1 < \mu_2 < \cdots < \mu_k$, which is exactly the constraint used in Richardson and Green (1997). However, this method is not always effective in overcoming the problem of removing the symmetry from the posterior distributions, see Stephens (2000b) for evidence. Also, a further drawback is that the results can be dependent on the constraint imposed, see Richardson and Green (1997). Therefore, they recommend that the sampler output should be post-processed. Other references that discuss the limits of identifiability constraints are Celeux et al. (2000) and Jasra et al. (2005)

Most other strategies have the common goal of minimising a loss function to find an effective labeling structure. These methods include those of Celeux (1998), Stephens (1997b, 2000b), Celeux et al. (2000) and Hurn et al. (2003).

The methods of Celeux (1998) and Stephens (1997b, 2000b) can be described as relabelling algorithms. They both try to achieve an optimal labelling structure by minimising a pre-defined loss function after the MCMC sample has been produced. Jasra et al. (2005) comments that these relabeling algorithms are essentially imposing identifiability constraints with the only difference to the above situation being that it is not done online. The other main approach, taken by Celeux et al. (2000) and Hurn et al. (2003), uses what can described as label-invariant loss functions. This method requires the estimation of the posterior expected loss where every quantity of interest has an associated loss function that has to be minimised. Examples of these loss functions can be seen in the aforementioned papers. For a more comprehensive summary of details of these techniques and the potential advantages/disadvantages for each see Chapter 4 in MacLachlan and Peel (2000) and Jasra et al. (2005).

3.2.5.1 Post-processing algorithm

This post-processing algorithm is a relabeling method for the sample of N allocation vectors from the allocation sampler. The method of Stephens (2000b) proposes using a relabeling algorithm that attempts to minimise the posterior expected loss under a class of loss functions to overcome the label-switching problem. The post-processing method to be defined here fits into the general framework of Stephens (2000b). Jasra et al. (2005) commented that the idea of Stephens (2000b) is that one defines a loss function on an action space A, and tries to minimise the loss by finding the most appropriate action a. The allocation sampler produces a slightly different setting to that of these previous

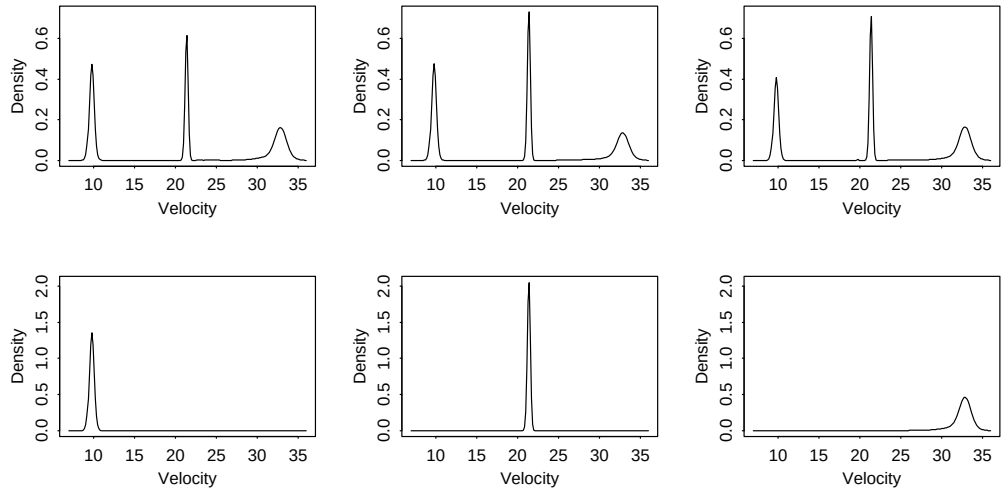

Figure 3.6: Galaxy data, marginal posterior distributions of the component means m_1, m_2 and m_3 in the mixture of normals model, conditional on $k = 3$. The top row of plots displays estimates of the posteriors based on the raw sample of g vectors from the allocation sampler. Bottom row contains estimates using the allocation vectors with re-assigned labels.

authors, because the parameters are no longer in the state space. However, the label-switching problem is still present, as shown in Figure 3.6.

In this case, an action can be defined as a permutation of the component labels σ: $\sigma g = (\sigma_{g_1}, \sigma_{g_2}, \ldots, \sigma_{g_n})$. Furthermore, the loss function can be thought of as a sum of the distances between all the allocation vectors,

$$Loss = \sum_{t=1}^{N-1} \sum_{s=t+1}^{N} D(\sigma^{(t)} g^{(t)}, \sigma^{(s)} g^{(s)}). \tag{3.37}$$

This loss function would then have to be minimised with respect to the sequence of permutations $\{\sigma^{(t)}, t = 1, \ldots, N\}$. The distance measure between two allocation

vectors g and g' can be defined as the number of coordinates where they differ:

$$D(g, g') = \sum_{i=1}^{n} I\{g_i \neq g_i'\}. \tag{3.38}$$

It can easily be shown that (3.38) defines a distance by satisfying the three conditions required for a distance measure. An approximate solution for this problem can be found by simplifying the problem into a sequence of optimization problems that only involve one permutation $\sigma^{(t)}$. This results in having now to solve a sequence of square assignment problems. The classic example of a square assignment problem is trying to assign P machines to P jobs, where there is an associated cost for each machine completing each job. This leads to the construction of a $P \times P$ cost matrix. An assignment of machines to jobs that minimises the total cost then has to be found. One algorithm used to solve these type of problems is called the Hungarian method. The code of Carpaneto and Toth (1980) uses the Hungarian method to solve the square assignment problem. The Hungarian method is a combinatorial optimization algorithm. The assignment problems that arise here are modelled by creating a cost matrix, where each element represents the cost of assigning component i to label j. The cost matrix is calculated by using the distances between the labellings of the allocation vectors. The method then tries to minimise the total cost of the current assignment of labels by finding an optimal permutation, $\sigma^{(t)}$.

Let $S = \{g^{(t)}, t = 1, \ldots, N\}$ be the sequence of all the sample allocation vectors produced from the allocation sampler and let $K = \{\tilde{k}^{(t)}, t = 1, \ldots, N\}$ be the sequence of the number of non-empty components contained in the vectors

of S. The procedure starts by firstly dealing with the vectors where $\tilde{k}^{(t)} = 2$ and proceeds by increasing $\tilde{k}^{(t)}$. Suppose $g^{(t)}$ has $\tilde{k}^{(t)} = j$. Then a cost matrix is produced by comparing the component label of each observation of $g^{(t)}$ with all the observations of all other allocation vectors where $\tilde{k}^{(t)} = j - 1$ or previously processed allocation vectors with $\tilde{k}^{(t)} = j$. This creates a $j \times j$ cost matrix that is minimised using the method of Carpaneto and Toth (1980) to yield a labelling structure $\sigma^{(t)}$ for $g^{(t)}$ that aligns with it all the other allocation vectors. After all allocation vectors with $\tilde{k}^{(t)} = j$ have been processed the procedure is repeated again for these vectors but this time all the vectors with $\tilde{k}^{(t)} = j$ or $j - 1$ are used in the construction of the cost matrix. The procedure could be implemented using all the available allocation vectors S. However, restricting the comparison of allocation vectors to vectors with the same number of non-empty components j, or $j - 1$ components, is done to improve the speed of the algorithm with the idea that the allocation vectors with $j-1$ components should already be alligned the allocation vectors containing $2, \cdots, j - 2$ non-empty components. There is little effect on the results of the relabelling algorithm with this restriction being imposed. The reason for comparing each allocation vector to allocation vectors with fewer components is so that the allignment of the labels is standard across the whole set S and not conditional on $\tilde{k}^{(t)}$.

A more formal definition of the algorithm is now given for processing vectors with the number of non-empty components $\tilde{k}^{(t)} = j$.

For $m = 1, \ldots, N$

If $\tilde{k}^{(m)} = j$ then, let $g = g^{(m)}$

Create cost matrix C of dimensions $j \times j$ that has elements defined by

$$C(j_1, j_2) = \sum_{g' \in B^{(t)}} \sum_{i=1}^{n} I\{g'_i \neq j_1, g_i = j_2\}$$

where $B^{(t)} = \{g^{(t)} \in S : \tilde{k}^{(t)} = j - 1 \cup (\tilde{k}^{(t)} = j \cap t < m), t = 1, \cdots, N\}$, $j_1 = 1, \ldots, j$ and $j_2 = 1, \ldots, j$.

Minimise the total cost $\sum_{h=1}^{\tilde{k}^{(t)}} C(h, \sigma_h^{(t)})$ using the method of Carpaneto and Toth (1980) and where= $\sigma_h^{(t)}$ is the permutation of label h on allocation vector t.

Reassign the labels of g according to the optimal permutation of the j labels.

Repeat the above procedure replacing $B^{(t)}$ with $D^{(t)} = \{g^{(t)} \in S, \tilde{k}^{(t)} = k - 1 \cup \tilde{k}^{(t)} = k, t = 1, \cdots, N\}$.

This algorithm is repeated for all values of $\tilde{k}^{(t)}$, to produce a new aligned set of allocation vectors in an increasing order starting with allocation vectors with 2 non-empty components. Now, a simple example to show the algorithm in practice

will be given for the following set of allocation vectors,

$$g_1 = (1, 1, 1, 2, 2, 2)$$
$$g_2 = (1, 1, 1, 3, 2, 2)$$
$$g_3 = (1, 1, 2, 3, 4, 2)$$
$$g_4 = (3, 3, 3, 2, 2, 1)$$
$$g_5 = (2, 2, 3, 3, 1, 1)$$
$$g_6 = (3, 3, 3, 2, 2, 1)$$

Now, suppose we are looking to find the optimal permutation of the labels for g_5. This requires the construction of a cost matrix by comparing g_5 to all the allocation vectors with 2 non-empty components, $\{g_1\}$, and also any allocation vectors with 3 non-empty components with an index less than 7, $\{g_4\}$. The cost matrix arising from the comparison of g_5 to these other 3 allocation vectors is

$$C = \begin{bmatrix} 5 & 2 & 4 \\ 1 & 6 & 4 \\ 6 & 4 & 4 \end{bmatrix}.$$

This cost matrix is then minimised using the method of Carpaneto and Toth (1980) and yields a minimum cost of 7 with the labels of g_5 being changed to $g_5 = (1, 1, 3, 3, 2, 2)$. The procedure would then continue on to g_6 and then return to look at g_4 before returning to g_5. This second treatment of g_5 would however be slightly different to the first because the allocation vector g_6 would be used in the construction of the cost matrix. For details on the performance of this procedure see Section 4.2.3.

Chapter 4

Simulation Study

In this chapter the allocation sampler introduced in Chapter 3 will be demonstrated in a large scale simulation study using randomly generated data. The main purpose of the simulation study is to test the allocation sampler in many different situations. Another study of how the allocation sampler performs can be found in Nobile and Fearnside (2007). In this paper a simulation study is carried out on a set of eight mixture models. The samples used were not random but were artificially produced representative samples for different sample sizes. However, in the simulation study reported here the samples used are randomly generated in order to mimick more realistic data situations.

4.1 Design of study

This simulation study has been designed so that a wide variety of features that are observed in real life datasets are observed in the examples. This will enable the study to examine where the allocation sampler performs best and also where

possible problems arise. The allocation sampler was applied to random samples of sizes 50, 200, 500 and 2000 from the 15 mixtures of univariate normals that appear in Marron and Wand (1992). For each of the 15 mixtures, 20 random samples were generated for the 4 different sample sizes. Therefore a total of 1200 data sets were analyzed. For full mixture descriptions and graphical displays, see Table 4.1 and Figure of the 15 mixtures. Marron and Wand (1992) also contains full details as to why each of these mixtures was chosen and what type of data they are supposed to represent.

The allocation sampler was coded in Fortran which produced output files that could then be used by R to produce the posterior results. A computer cluster was used to execute the allocation sampler for this study due to the high volume of data sets. The computer cluster is composed of a headnode and 60 compute nodes each consisting of 2 opteron 248 processors and 2GB RAM connected together over a gigabit ethernet. An approximation for the total amount of processor time required run the allocation sampler for this simulation study is 775 hours. This was completed within 1 month with the help of the computer cluster.

4.1.1 Allocation Sampler procedure

The allocation sampler was implemented in the same way for each of the 1200 datasets. Firstly, a preliminary run was executed to calculate the hyperparameter values τ and δ, and also to enable a thinning parameter, Δ, to be chosen. The starting hyperparameter values for this run were $\alpha = 1$, $\mu = \bar{x}$, $\tau = 0.5$, $\gamma = 2$ and $\delta = 0.5s_x^2$ where $\bar{x}$ is the sample mean and s_x^2 is the sample variance. The sampler was started from $k = 1$ and had a burn-in of 10000 iterations preceding another

500000 iterations. A thinning parameter, $\Delta = 10$, was used to produce a sample of 50000. The hyperparameters τ and δ were then estimated according to the procedure in Section 3.2.4. The thinning parameter was then chosen by looking at the autocorrelation of the sampled k's from the preliminary run. It was chosen so that the thinning value is likely to achieve a lag-1 autocorrelation of 0.7 in the sampled k's. Next, a run of the allocation sampler was performed comprising 10000Δ moves, plus 1000Δ moves of burn-in using fixed hyperparameters. This run used, as a starting position, the final allocation vector g and k from the preliminary run. This was done to help the convergence of the Markov chain.

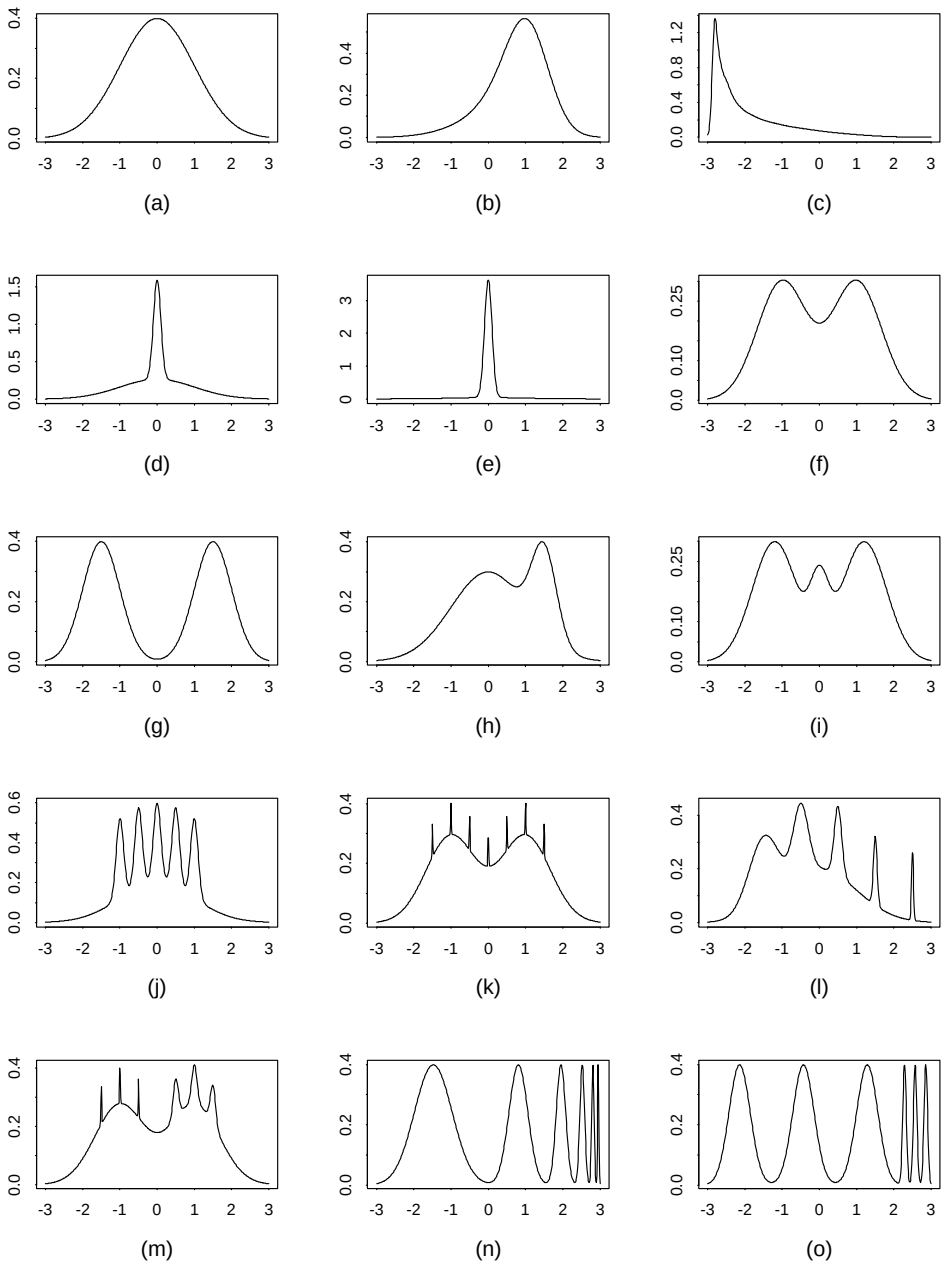

Figure 4.1: Density functions of mixtures of univariate normal distributions from Marron and Wand (1992). See Table (4.1) for the parameters of these models.

	Name	k_{TRUE}	Mixture Density
(a)	Gaussian	1	$N(0,1)$
(b)	Skewed Unimodal	3	$\frac{1}{5}N(0,1) + \frac{1}{5}N(\frac{1}{2},(\frac{2}{3})^2) + \frac{3}{5}N(\frac{13}{12},(\frac{5}{9})^2)$
(c)	Strongly Skewed	8	$\sum_{l=0}^{7} \frac{1}{8}N(3\{(\frac{2}{3})^l - 1\},(\frac{2}{3})^{2l})$
(d)	Kurtotic Unimodal	2	$\frac{2}{3}N(0,1) + \frac{1}{3}N(0,(\frac{1}{10})^2)$
(e)	Outlier	2	$\frac{1}{10}N(0,1) + \frac{9}{10}N(0,(\frac{1}{10})^2)$
(f)	Bimodal	2	$\frac{1}{2}N(-1,(\frac{2}{3})^2) + \frac{1}{2}N(1,(\frac{2}{3})^2)$
(g)	Separated Bimodal	2	$\frac{1}{2}N(-\frac{3}{2},(\frac{1}{2})^2) + \frac{1}{2}N(\frac{3}{2},(\frac{1}{2})^2)$
(h)	Skewed Bimodal	2	$\frac{3}{4}N(0,1) + \frac{1}{4}N(\frac{3}{2},(\frac{1}{3})^2)$
(i)	Trimodal	3	$\frac{9}{20}N(-\frac{6}{5},(\frac{3}{5})^2) + \frac{9}{20}N(\frac{6}{5},(\frac{3}{5})^2) + \frac{1}{10}N(0,(\frac{1}{4})^2)$
(j)	Claw	6	$\frac{1}{2}N(0,1) + \sum_{l=0}^{4} \frac{1}{10}N(l/2-1,(\frac{1}{10})^2)$
(k)	Double Claw	9	$\frac{49}{100}N(-1,(\frac{2}{3})^2) + \frac{49}{100}N(1,(\frac{2}{3})^2) + \sum_{l=0}^{6} \frac{1}{350}N((l-3)/2,(\frac{1}{100})^2)$
(l)	Asymmetric Claw	6	$\frac{1}{2}N(0,1) + \sum_{l=-2}^{2} (2^{1-l}/31)N(l+\frac{1}{2},(2^{-l}/10)^2)$
(m)	Asymmetric Double Claw	8	$\sum_{l=0}^{1} \frac{46}{100}N(2l-1,(\frac{2}{3})^2) + \sum_{l=1}^{3} \frac{1}{300}N(-l/2,(\frac{1}{100})^2) + \sum_{l=1}^{3} \frac{7}{300}N(l/2,(\frac{7}{100})^2)$
(n)	Smooth Comb	6	$\sum_{l=0}^{5} (2^{5-l}/63)N(\{65 - 96(\frac{1}{2})^l\}/2l,(\frac{32}{63})^2/2^{2l})$
(o)	Discrete Comb	6	$\sum_{l=0}^{2} \frac{2}{7}N((12l-15)/7,(\frac{2}{7})^2) + \sum_{l=8}^{10} \frac{1}{21}N(2l/7,(\frac{1}{21})^2)$

Table 4.1: Parameters for the 15 mixtures of univariate normal distributions as displayed in Figure 4.1

4.2 Sampler Performance

4.2.1 Posterior of k

The question of interest about the posterior of k is whether the modal k from the allocation sampler equals that of the true number of components in the model, k_{TRUE}, shown in Table 4.1. The posterior distribution of k is found by equation (2.28). Box-plot summaries of the distribution of the estimated posterior probabilities, $Pr[K = k]$, over the 20 random samples for a given sample size are displayed in Figures (4.2) - (4.16). It can be seen that in 10 out of the 15 models the median of $Pr[K = k_{TRUE}]$ is greater than 0.5 using a sample of 2000. Also, in 6 out these 10 cases, {(a),(d),(e),(f),(g),(h)}, the allocation sampler is able to produce a modal k equal to k_{TRUE} from a sample of just 50. It is obvious from all the plots that as the sample size increases the modal value $\pi(k|x)$ tends towards k_{TRUE}. Also, one would expect that, if the components overlap each other, then it will be more difficult for the sampler to extract the correct number of components. This can certainly be seen in Figures (4.3), (4.4), (4.12) and (4.14). The sampler has no problem picking the correct number of components when the sample has come from a model where the components are well separated, for example in model (g). However, in models {(b),(c),(k),(m)}, where there is significant overlap of the components, the modal k never reaches k_{TRUE} even with a sample of 2000. Another problem for the allocation sampler, when estimating the posterior of k, is that there are sometimes only relatively few observations arising from a certain component. An example of this can be seen in model (k), where the 7 spikes on the underlying bimodal distribution each have a weight of $\frac{1}{350}$.

Hence, the random samples may sometimes not even contain any observations from some of these components, and this leads to the underestimation of k.

4.2.2 Posterior Predictive Distributions

This section discusses how well the posterior predictive distributions model the the mixture distributions from Table 4.1 from the randomly generated samples. The posterior predictive distributions are displayed in Figures (4.2) - (4.16). These distributions were calculated by evaluating the mean of (2.48) for the 20 random samples over the range of values $(-3, 3)$. The expression (2.48) is the posterior predictive distribution averaged over both k and g. The figures show that, as the sample size increases, the posterior predictive distributions move closer to the true density. Furthermore, they indicate that the allocation sampler manages to model all of the main features in 13 out of the 15 models when using a sample size of 2000. The only 2 models where the main features are not modeled are in models (k) and (m). These two models both have sharp spikes as features that are never realised in the posterior predictive distributions for the same reason that k_{TRUE} was not found in these cases. When a sample of 50 is used, the main features are only picked up in 3 cases, namely {(a),(b),(e)}. A final point to note is that in Figures (4.3) and (4.4), for mixtures (b) and (c), the posterior predictive models the mixture very well even though k_{TRUE} is not favoured by the allocation sampler.

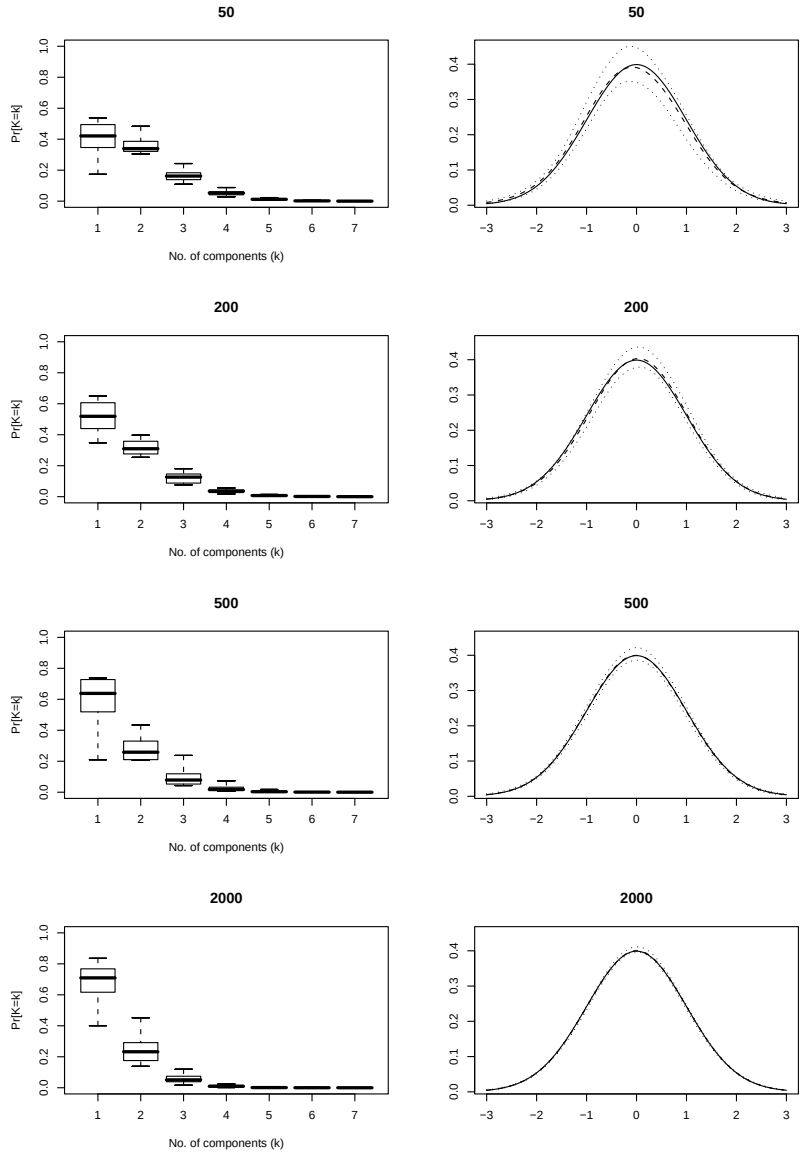

Figure 4.2: (a) Gaussian, $k_{TRUE} = 1$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

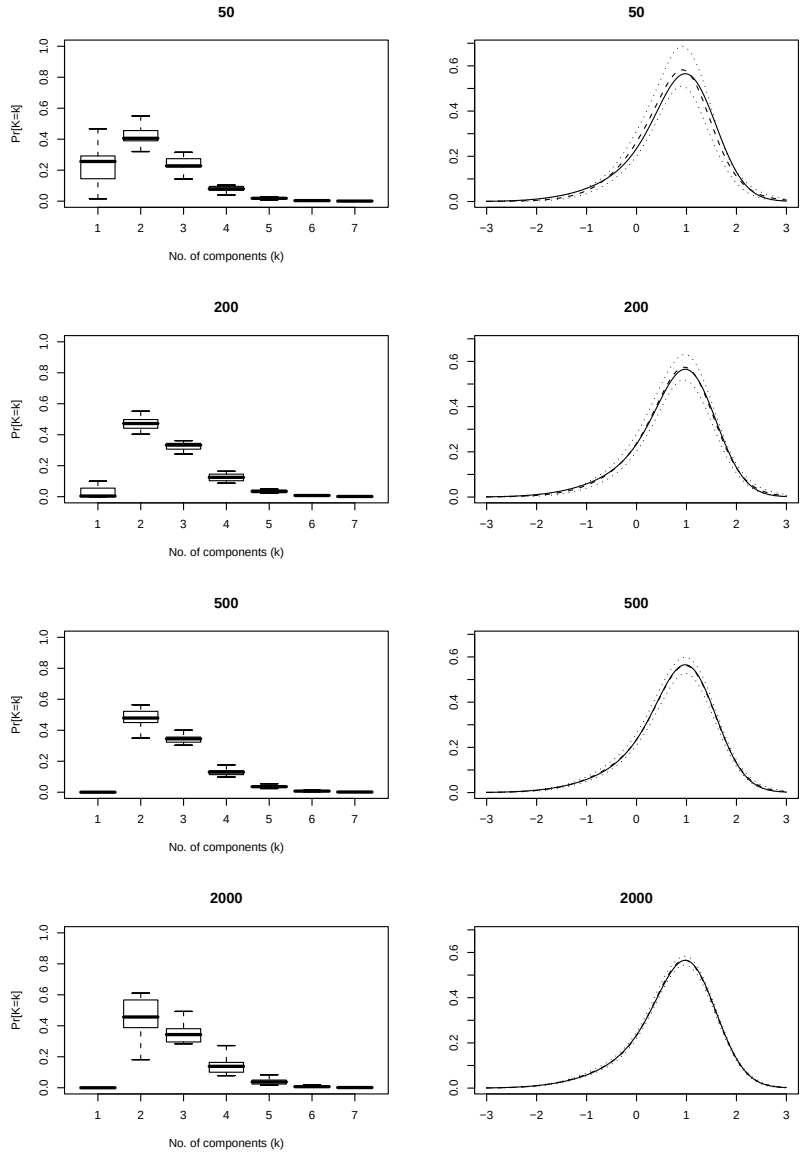

Figure 4.3: (b) Skewed Unimodal, $k_{TRUE} = 3$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

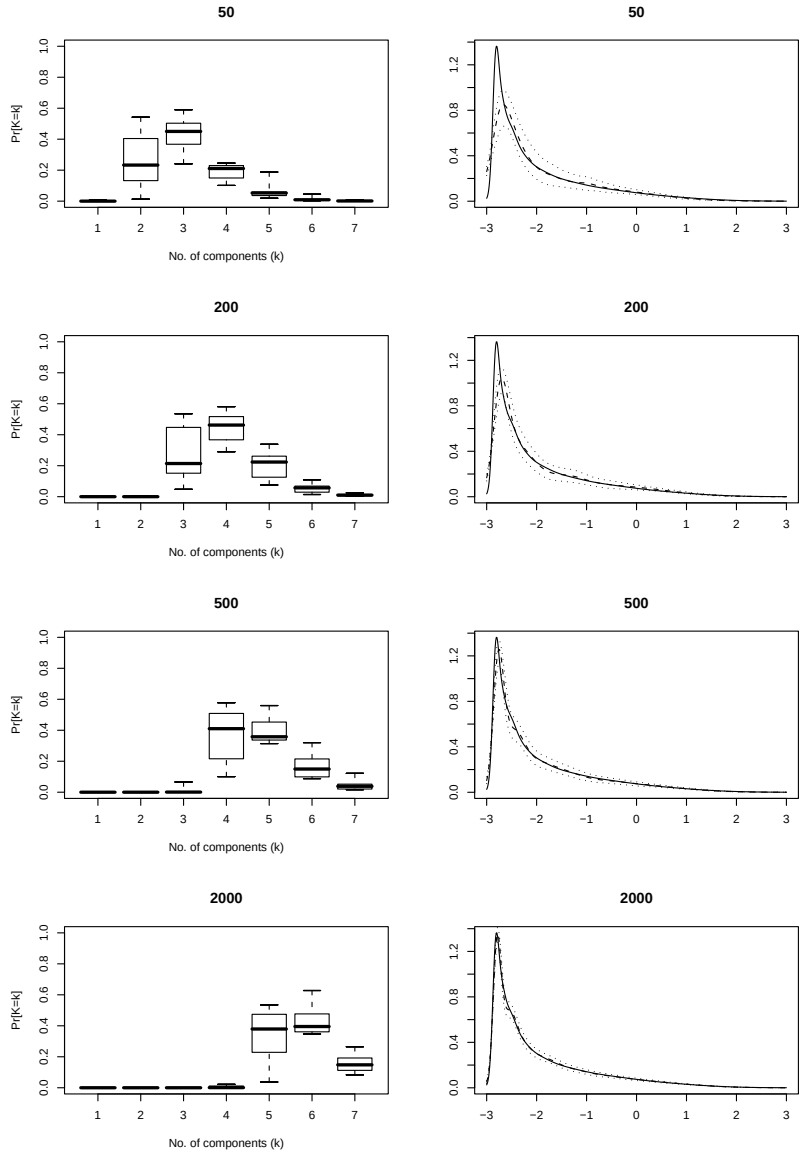

Figure 4.4: (c) Strongly Skewed, $k_{TRUE} = 8$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

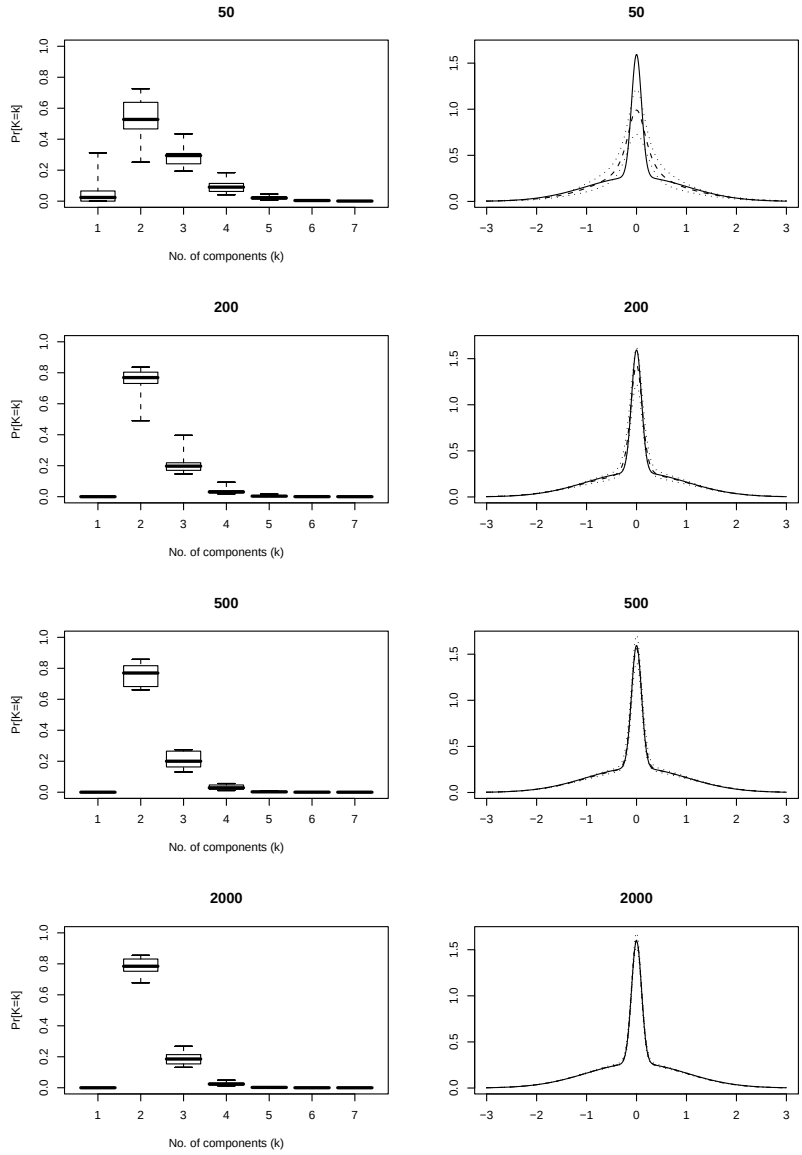

Figure 4.5: (d) Kurtotic Unimodal, $k_{TRUE} = 2$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

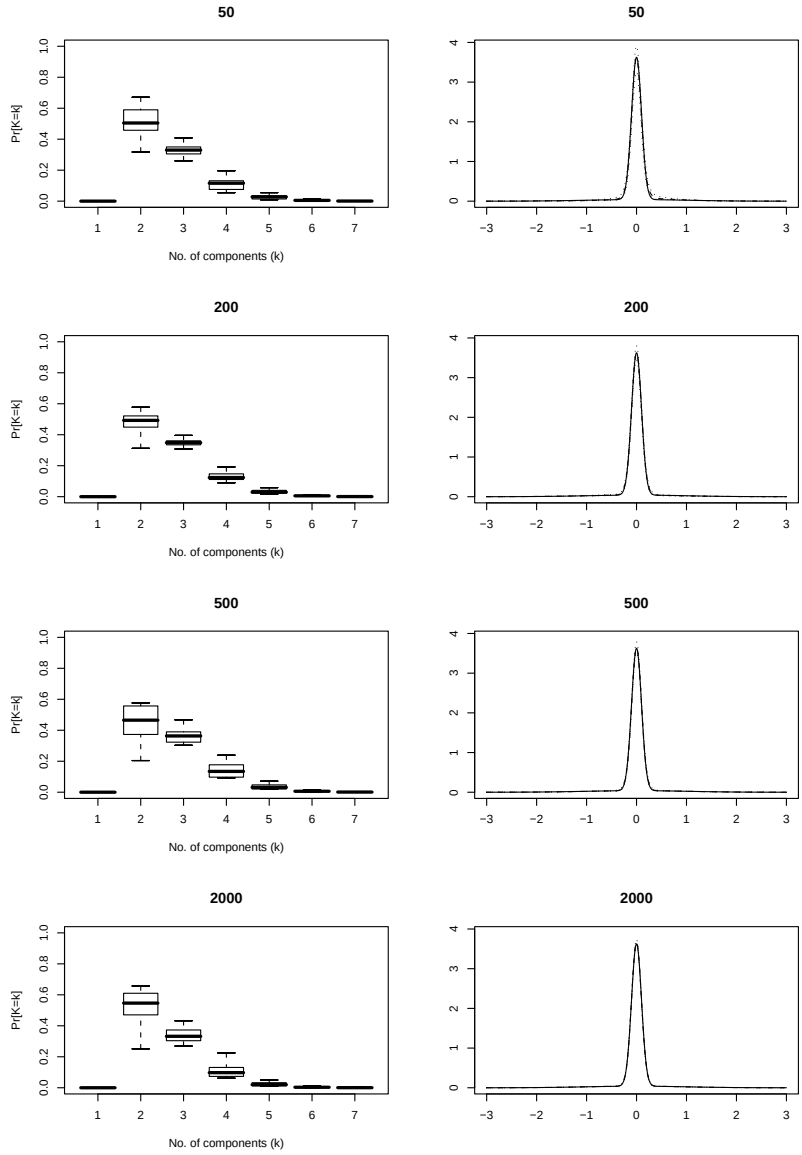

Figure 4.6: (e) Outlier, $k_{TRUE} = 2$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

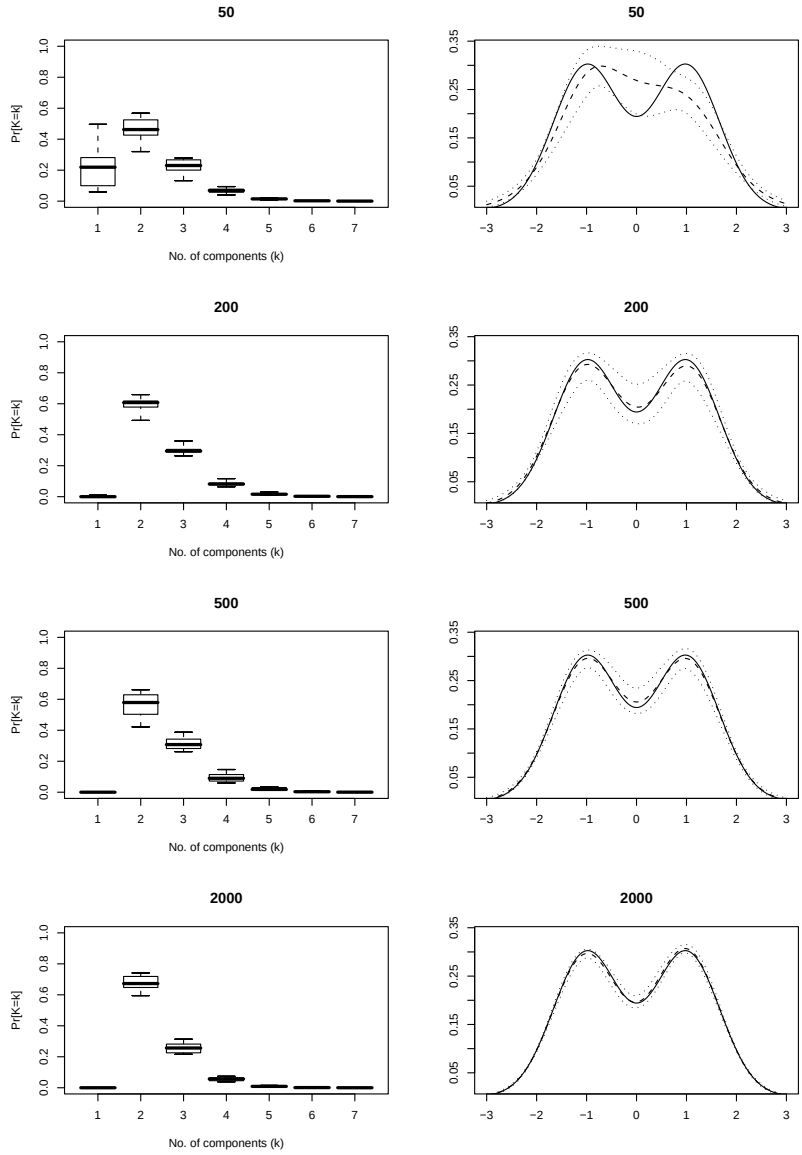

Figure 4.7: (f) Bimodal, $k_{TRUE} = 2$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

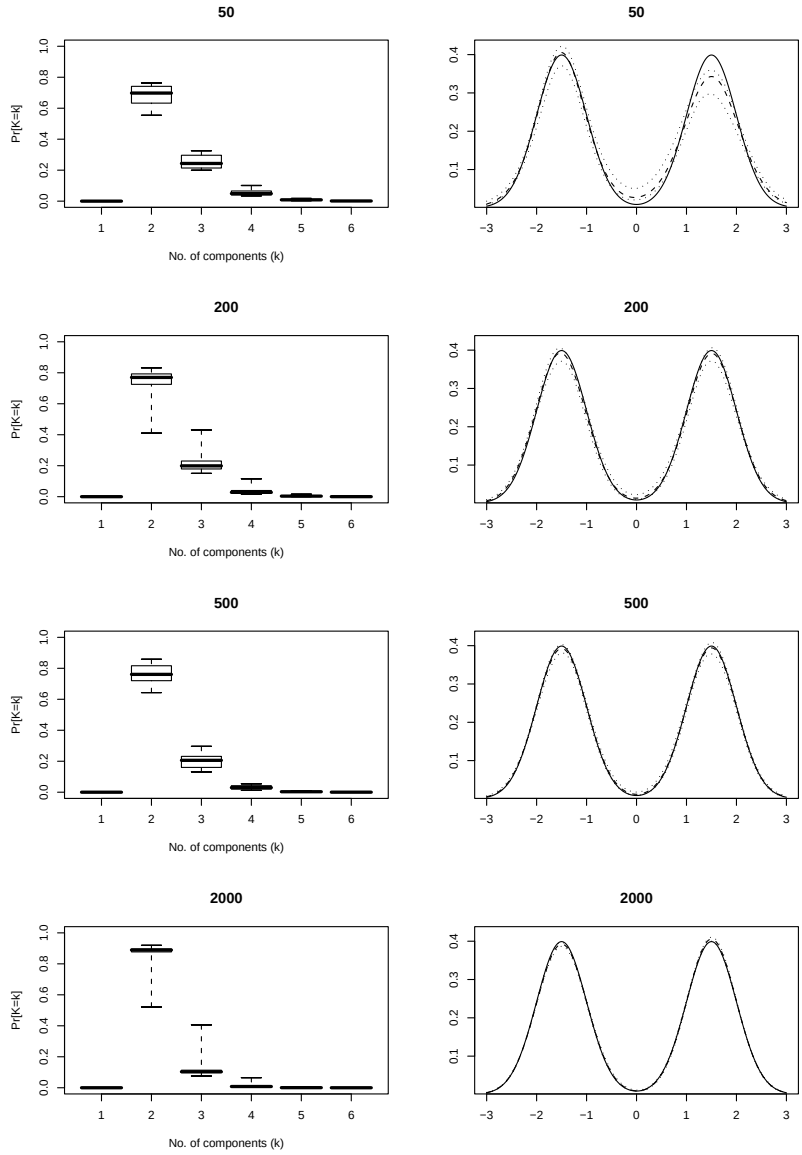

Figure 4.8: (g) Separated Bimodal, $k_{TRUE} = 2$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

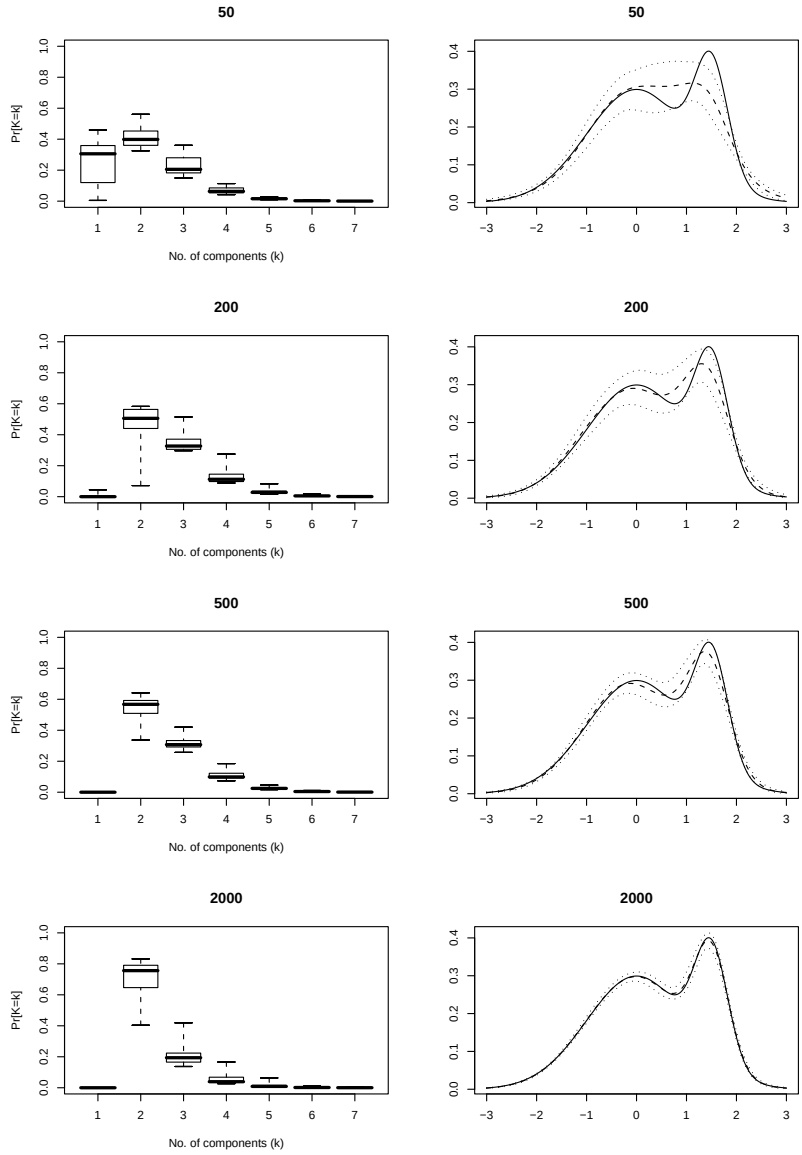

Figure 4.9: (h) Skewed Bimodal, $k_{TRUE} = 2$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

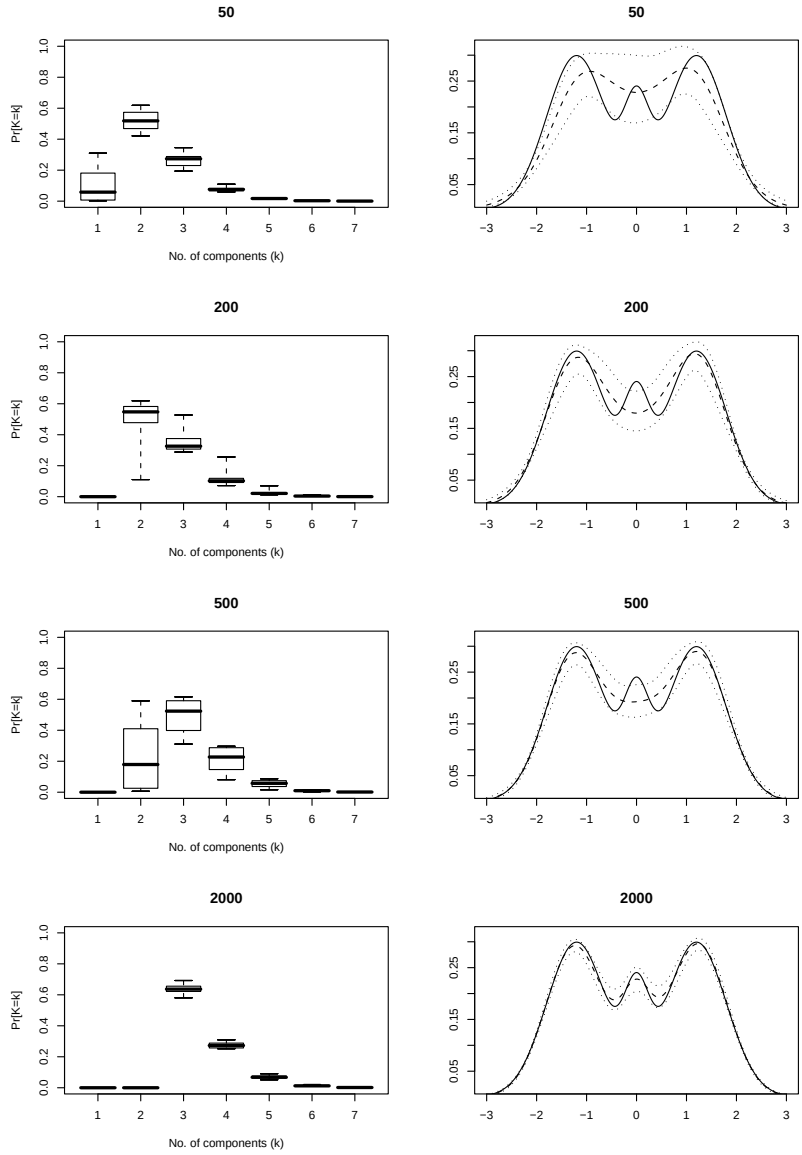

Figure 4.10: (i) Trimodal, $k_{TRUE} = 3$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

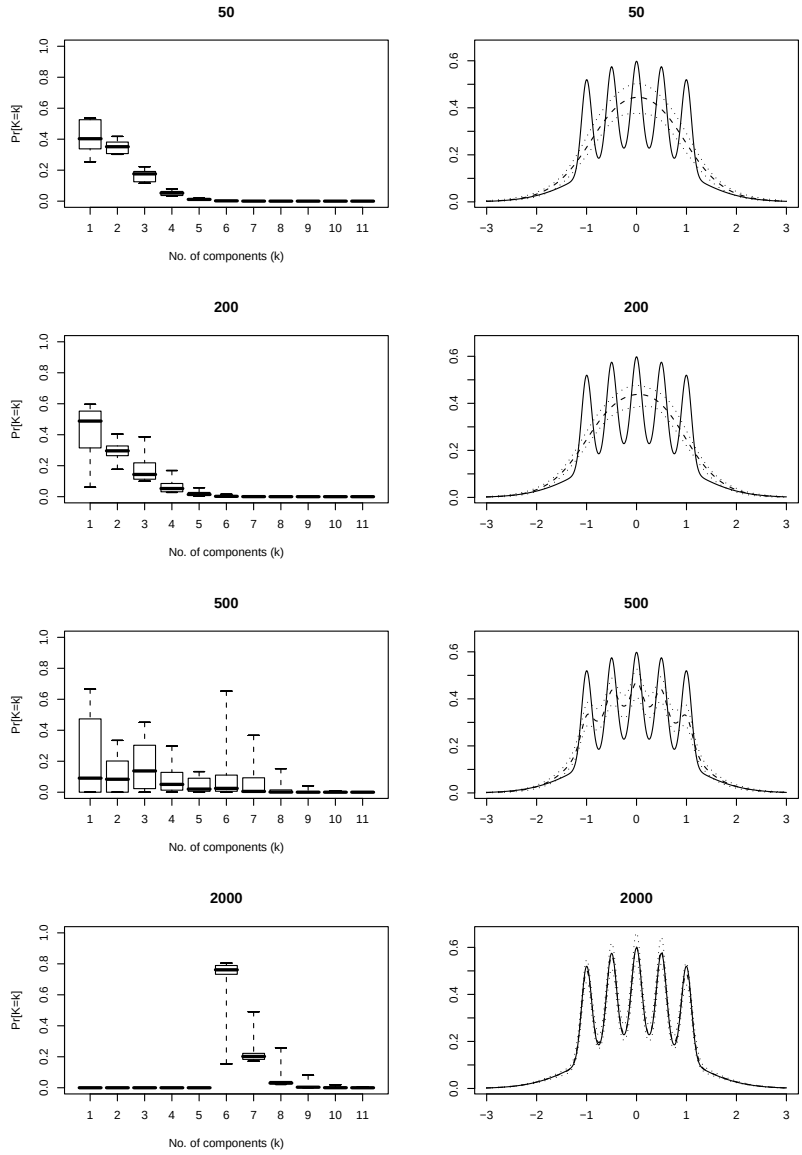

Figure 4.11: (j) Claw, $k_{TRUE} = 6$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

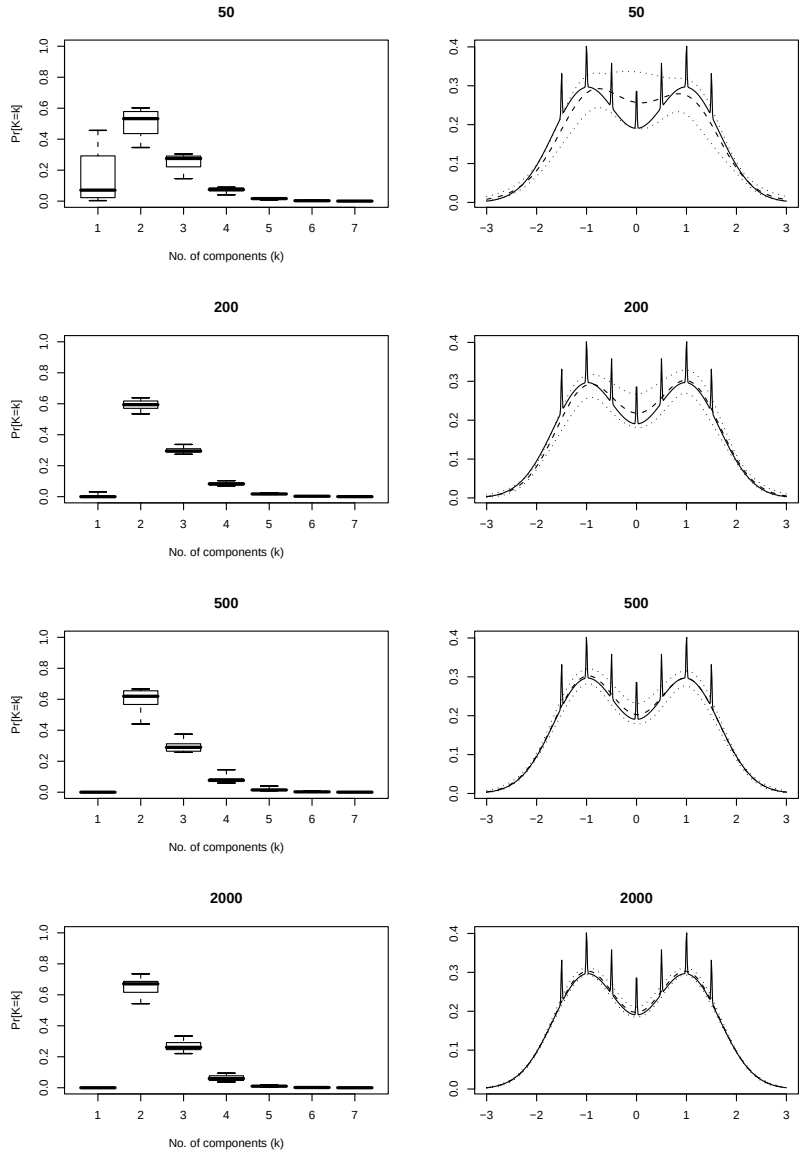

Figure 4.12: (k) Double Claw, $k_{TRUE} = 9$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

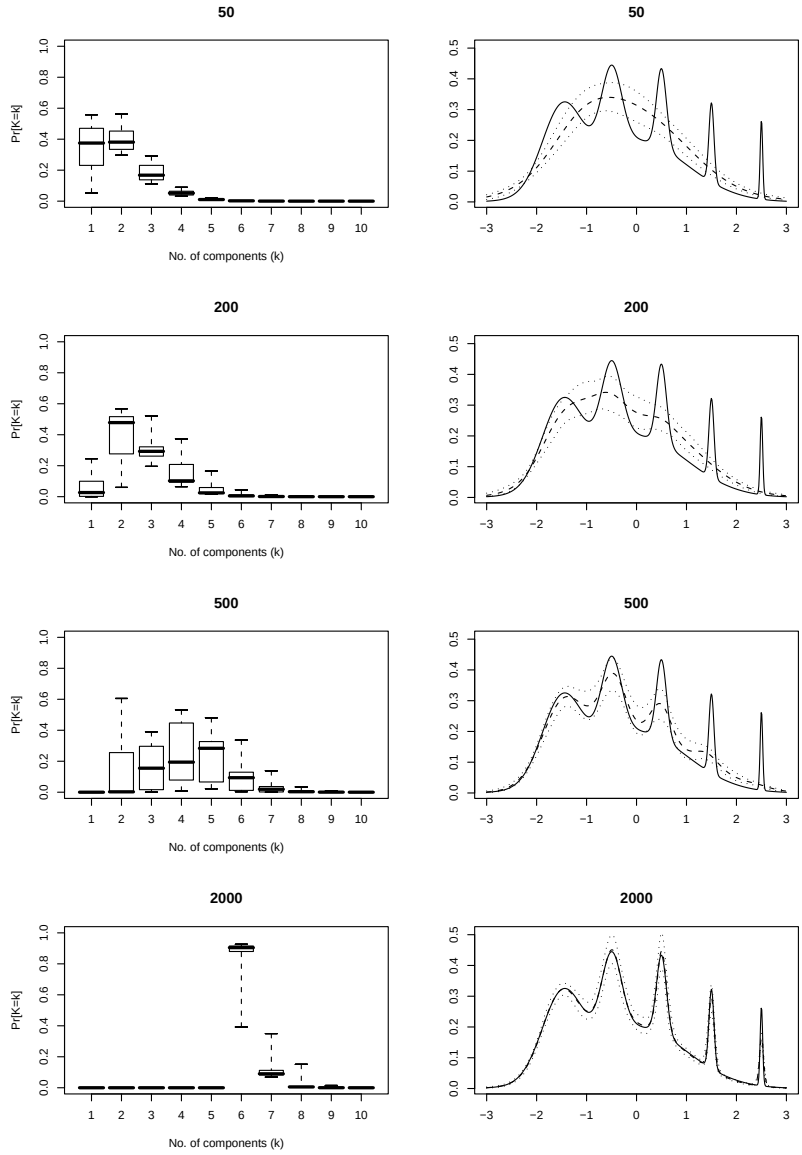

Figure 4.13: (l) Asymmetric Claw, $k_{TRUE} = 6$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

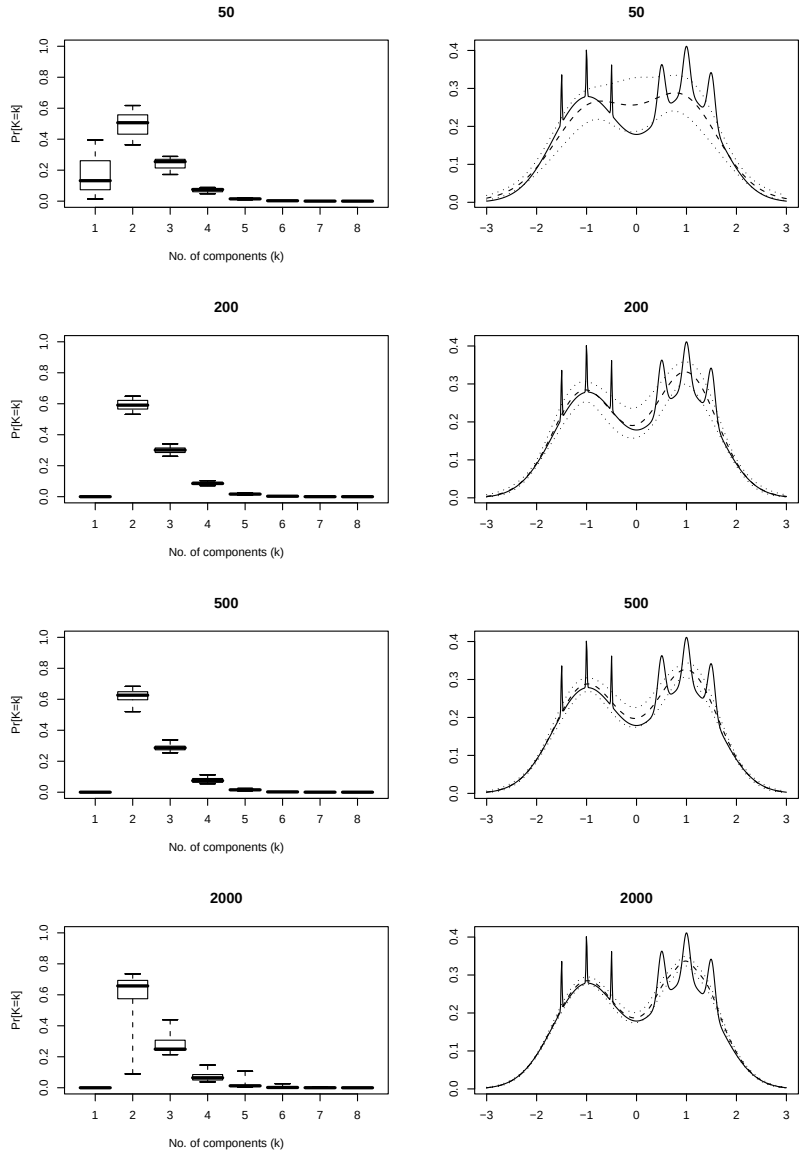

Figure 4.14: (m) Asymmetric Double Claw, $k_{TRUE} = 8$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

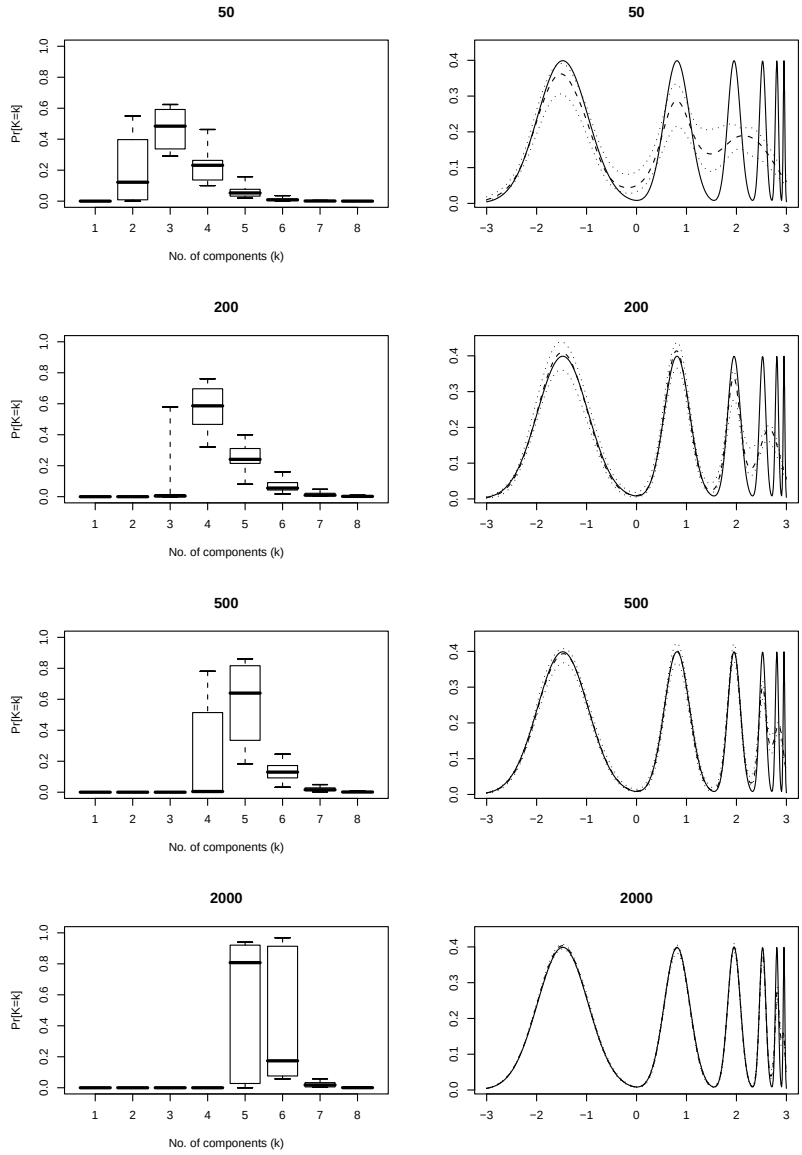

Figure 4.15: (n) Smooth Comb, $k_{TRUE} = 6$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

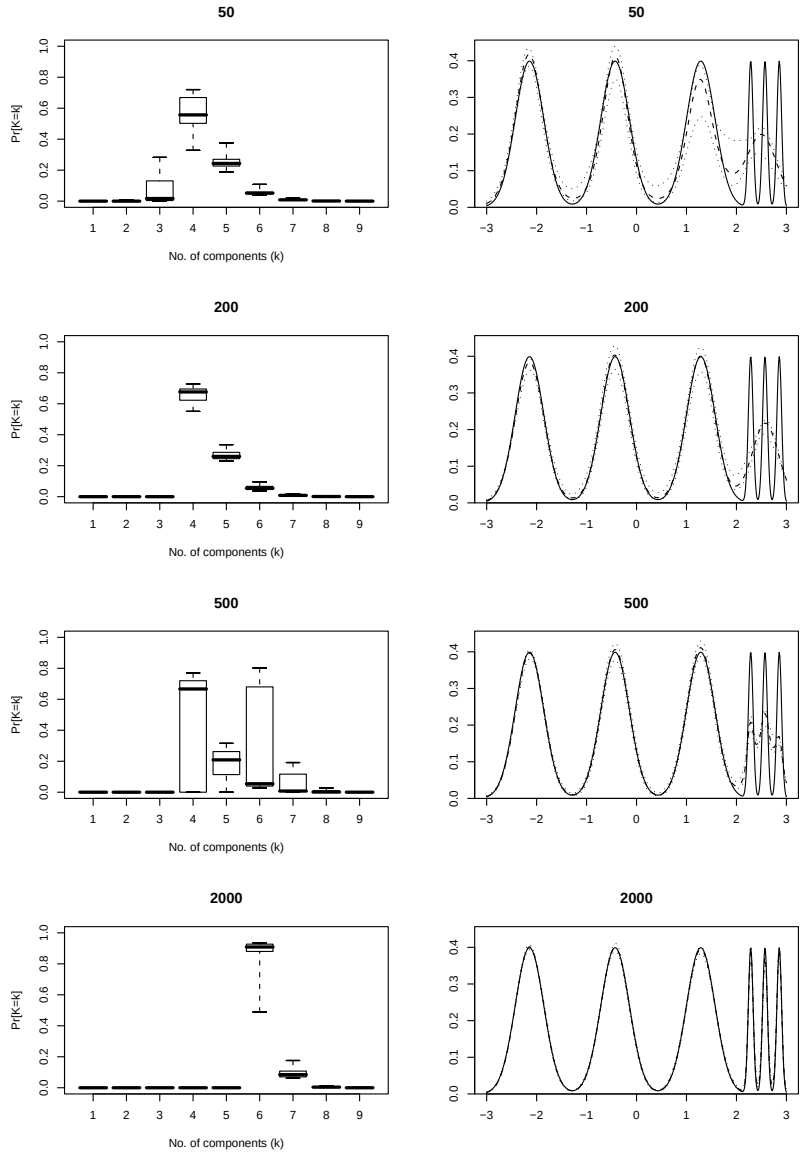

Figure 4.16: (o) Discrete Comb, $k_{TRUE} = 6$: The left-hand column displays the posterior of k by using boxplots of the posterior probabilities, $Pr[K = k]$. The whiskers of the plots stretch to the 5% and 95% quantiles. The right-hand column of the plot displays the posterior predictive density calculated by averaging over the 20 samples for each sample size. The full line shows the true density, the dashed line shows the posterior predictive and the dotted lines show 95% confidence bands for the posterior predictive density.

4.2.3 Parametric Inference

The aim of this section is to investigate whether the allocation sampler can recover parameter estimates that are close to the real parameter values given in Table 4.1. For parameter estimation to be possible, the label-switching problem described in Section 3.2.5 has to be dealt with. The post-processing method detailed in Section 3.2.5.1 was used here to try and remove the symmetry from the posterior parameter densities. Thus, this section also helps give an indication about how well the post-processing algorithm from Section 3.2.5.1 performs. The results reported in Tables (4.3) - (4.11) are based on a single run of the allocation sampler using a sample of size $n = 2000$. Only mixtures where the highest posterior probability of k is equal to k_{TRUE} are analysed, therefore allowing direct comparison of the parameters in the model. The number of allocation vectors used for each of the mixtures is given in Table (4.2). This was only carried out for 9 of the 15 mixtures of normals, see Figures (4.2) - (4.16). Note that the Gaussian case (a) was also omitted because of its simplicity. Running this post-processing algorithm, which was again coded in Fortran, required anything from 4 hours up to 9 hours to complete. As would be expected the run times increase for models with a higher number of components.

In each of Tables (4.3) - (4.11) the label-switching problem is evident if one looks at the raw output. The estimates are essentially equal and also have fairly high standard deviations. However, when the output has been post-processed using the method detailed in Section 3.2.5.1, the estimates appear to be more meaningful and the standard deviations of these estimates reduce dramatically compared to the raw output. Therefore, the post-processing algorithm is working

effectively in reducing the symmetry of the posterior parameter densities. When comparing the post-processed parameter estimates to the real values recorded in Table (4.1), relatively few of the real values lie further than one standard deviation from the estimate. In mixtures {(d),(e),(f)} the real values lie within one standard deviation for all the parameters in the model. In the remaining models all the real values of the mixture weights, means and standard deviations are within two standard deviations of the estimates with two exceptions. They are the standard deviations for component 6 in model (l) and component 4 in model (o). Therefore, one is able to conclude that the allocation sampler performs well in dealing with the label-switching problem and estimating the parameters of the model.

Mixture	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(l)	(o)
No. of vectors	8016	5998	6850	8911	6779	6924	7958	9120	9271

Table 4.2: The number of allocation vectors used to calculate the parameter estimates in Tables (4.3) - (4.11).

Parameters	True	Raw output	Post-processed
λ_1	0.67	0.50 ± 0.16	0.66 ± 0.01
λ_2	0.33	0.50 ± 0.16	0.34 ± 0.01
m_1	0	-0.02 ± 0.23	-0.03 ± 0.21
m_2	0	0.04 ± 0.24	0.05 ± 0.27
$\sqrt{r_1}$	1	5.68 ± 4.74	1.02 ± 0.02
$\sqrt{r_2}$	10	5.83 ± 4.75	10.49 ± 0.51

Table 4.3: Estimates of the parameters in mixture (d) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.1	0.49 ± 0.40	0.10 ± 0.01
λ_2	0.9	0.51 ± 0.40	0.90 ± 0.01
m_1	0	-0.01 ± 0.11	0.00 ± 0.13
m_2	0	0.00 ± 0.11	-0.01 ± 0.11
$\sqrt{r_1}$	1	5.51 ± 4.59	1.02 ± 0.06
$\sqrt{r_2}$	10	5.70 ± 4.59	10.19 ± 0.19

Table 4.4: Estimates of the parameters in mixture (e) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.5	0.50 ± 0.02	0.50 ± 0.02
λ_2	0.5	0.50 ± 0.02	0.50 ± 0.02
m_1	-1	0.03 ± 0.99	-0.96 ± 0.04
m_2	1	0.02 ± 0.99	1.02 ± 0.04
$\sqrt{r_1}$	1.5	1.50 ± 0.07	1.47 ± 0.06
$\sqrt{r_2}$	1.5	1.50 ± 0.07	1.52 ± 0.06

Table 4.5: Estimates of the parameters in mixture (f) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.5	0.50 ± 0.01	0.51 ± 0.01
λ_2	0.5	0.50 ± 0.01	0.49 ± 0.01
m_1	-1.5	-0.09 ± 1.49	-1.48 ± 0.01
m_2	1.5	0.11 ± 1.49	1.50 ± 0.01
$\sqrt{r_1}$	2	2.01 ± 0.10	2.10 ± 0.05
$\sqrt{r_2}$	2	2.01 ± 0.10	1.92 ± 0.05

Table 4.6: Estimates of the parameters in mixture (g) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.75	0.49 ± 0.24	0.74 ± 0.03
λ_2	0.25	0.51 ± 0.24	0.26 ± 0.03
m_1	0	0.74 ± 0.75	-0.02 ± 0.05
m_2	1.5	0.70 ± 0.75	1.47 ± 0.02
$\sqrt{r_1}$	1	1.91 ± 0.88	1.02 ± 0.03
$\sqrt{r_2}$	3	1.87 ± 0.89	2.76 ± 0.23

Table 4.7: Estimates of the parameters in mixture (h) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.45	0.33 ± 0.15	0.45 ± 0.03
λ_2	0.45	0.33 ± 0.15	0.42 ± 0.02
λ_3	0.10	0.34 ± 0.15	0.12 ± 0.04
m_1	-1.2	-0.12 ± 1.01	-1.26 ± 0.04
m_2	1.2	0.02 ± 1.01	1.20 ± 0.04
m_3	0	0.01 ± 1.01	-0.02 ± 0.06
$\sqrt{r_1}$	1.67	2.27 ± 0.87	1.67 ± 0.08
$\sqrt{r_2}$	1.67	2.28 ± 0.88	1.76 ± 0.09
$\sqrt{r_3}$	4	2.25 ± 0.87	3.36 ± 0.68

Table 4.8: Estimates of the parameters in mixture (i) with a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.5	0.16 ± 0.11	0.43 ± 0.04
λ_2	0.1	0.18 ± 0.13	0.11 ± 0.02
λ_3	0.1	0.16 ± 0.11	0.12 ± 0.01
λ_4	0.1	0.16 ± 0.11	0.11 ± 0.01
λ_5	0.1	0.16 ± 0.11	0.12 ± 0.01
λ_6	0.1	0.19 ± 0.11	0.13 ± 0.01
m_1	0	0.07 ± 0.61	0.01 ± 0.01
m_2	-1.0	-0.02 ± 0.63	-0.99 ± 0.01
m_3	-0.5	0.03 ± 0.68	-0.48 ± 0.01
m_4	0	-0.02 ± 0.63	0.06 ± 0.04
m_5	0.5	-0.06 ± 0.60	0.51 ± 0.01
m_6	1.0	0.09 ± 0.59	0.98 ± 0.01
$\sqrt{r_1}$	1	8.14 ± 3.03	0.96 ± 0.04
$\sqrt{r_2}$	10	7.71 ± 3.46	9.77 ± 0.97
$\sqrt{r_3}$	10	8.01 ± 3.10	9.09 ± 0.92
$\sqrt{r_4}$	10	8.11 ± 3.11	9.39 ± 0.99
$\sqrt{r_5}$	10	7.91 ± 3.11	9.88 ± 0.95
$\sqrt{r_6}$	10	7.32 ± 3.60	8.13 ± 0.76

Table 4.9: Estimates of the parameters in mixture (j) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.50	0.18 ± 0.17	0.49 ± 0.05
λ_2	0.26	0.15 ± 0.17	0.22 ± 0.17
λ_3	0.13	0.16 ± 0.17	0.17 ± 0.04
λ_4	0.06	0.21 ± 0.17	0.06 ± 0.01
λ_5	0.03	0.17 ± 0.17	0.04 ± 0.01
λ_6	0.02	0.16 ± 0.17	0.01 ± 0.003
m_1	0	-0.06 ± 1.04	-0.07 ± 0.07
m_2	-1.5	0.41 ± 1.44	-1.52 ± 0.05
m_3	-0.5	0.67 ± 1.39	-0.52 ± 0.04
m_4	0.5	0.32 ± 1.34	0.49 ± 0.01
m_5	1.5	0.41 ± 1.21	1.48 ± 0.01
m_6	2.5	0.57 ± 1.31	2.49 ± 0.01
$\sqrt{r_1}$	1	8.60 ± 6.96	0.97 ± 0.03
$\sqrt{r_2}$	2.5	9.29 ± 7.26	2.78 ± 0.29
$\sqrt{r_3}$	5	10.04 ± 7.33	4.14 ± 0.95
$\sqrt{r_4}$	10	6.35 ± 5.52	10.69 ± 1.50
$\sqrt{r_5}$	20	9.68 ± 7.22	20.21 ± 2.88
$\sqrt{r_6}$	40	8.50 ± 6.73	13.69 ± 1.92

Table 4.10: Estimates of the parameters in mixture (l) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

Parameters	True	Raw output	Post-processed
λ_1	0.29	0.17 ± 0.12	0.27 ± 0.01
λ_2	0.29	0.17 ± 0.12	0.30 ± 0.01
λ_3	0.29	0.18 ± 0.12	0.28 ± 0.01
λ_4	0.05	0.16 ± 0.12	0.05 ± 0.005
λ_5	0.05	0.17 ± 0.12	0.05 ± 0.005
λ_6	0.05	0.15 ± 0.11	0.05 ± 0.005
m_1	-2.14	0.97 ± 1.87	-2.14 ± 0.01
m_2	-0.43	0.97 ± 1.77	-0.42 ± 0.01
m_3	1.29	0.99 ± 1.87	1.28 ± 0.01
m_4	2.29	1.20 ± 1.72	2.28 ± 0.005
m_5	2.57	1.42 ± 1.62	2.57 ± 0.001
m_6	2.86	1.11 ± 1.83	2.85 ± 0.005
$\sqrt{r_1}$	3.5	11.85 ± 8.53	3.58 ± 0.12
$\sqrt{r_2}$	3.5	11.26 ± 8.31	3.63 ± 0.11
$\sqrt{r_3}$	3.5	10.87 ± 8.13	3.38 ± 0.11
$\sqrt{r_4}$	21	11.84 ± 8.09	18.01 ± 1.35
$\sqrt{r_5}$	21	11.45 ± 8.24	19.28 ± 1.42
$\sqrt{r_6}$	21	12.45 ± 8.01	21.85 ± 1.64

Table 4.11: Estimates of the parameters of mixture (o) using a sample size of $n = 2000$ and conditional on the true value of k as reported in Table (4.1). The column "Raw output" contains means and standard deviations of the marginal posteriors based on the sampler's raw output. The same quantities, after re-assigning the labels, are shown in column "Post-processed". The true parameter values are reported in column "True".

4.2.4 Mixing and Convergence Properties

An important aspect of any MCMC sampler is that the Markov chain should mix well, and it should also converge to the target distribution at some time-point during the burn-in period. Firstly, if one is to assess the mixing properties of the allocation sampler, then one requires to calculate the acceptance rates for each of the moves that are proposed. However, note that the Gibbs move is always accepted and therefore is improving the mixing of the chain when k is fixed.

The acceptance rates for all the Metropolis-Hastings moves have been calculated for every run of the allocation sampler and are summarised in Figures 4.17 and 4.18. These graphs show an overall general trend, with a few exceptions, that, as the sample size increases, the acceptance rates of the moves decrease. The acceptance rates start off at values of approximately 10% for a sample size of 50, but, when the sample size increases to 2000, these rates fall to as low as 0.01% in certain cases. This trend can be accounted for by the fact that as the sample size increases the information about the underlying model becomes stronger, and hence the allocation sampler finds it harder to move around the state space. Furthermore, it can be noted that the acceptance rates for the third fixed-k Metropolis-Hastings move is higher than for the other two moves in the majority of cases. The acceptance rates of the absorption/ejection move follow the same general trend as that for the fixed-k moves; i.e. as the sample size increases the acceptance rate decreases. However, these moves seem to have slightly higher acceptance rates on average than the fixed k moves.

Table 4.12 shows the median thinning parameters found from the preliminary runs using the procedure from Section 3.2.4. These thinning parameters give an

indication of how well the allocation sampler is mixing. The higher the thinning parameter used, the harder it is for the sampler to able to move around the state space of $\{k, g\}$. This table shows similar trends to the acceptance probability graphs in that as the sample size increases the thinning parameter also increases, meaning that the sampler is finding it harder to mix efficiently. The table also shows the median run times in seconds. It is obvious that as the sample size increases the run time increases accordingly. Furthermore, this table shows a general trend that the more complicated the model, the more thinning is required and hence longer run time.

Sample size		Thinning Δ				Run Time			
		50	200	500	2000	50	200	500	2000
Mixture	(a)	10	20	30	35	1	16	53	313
	(b)	10	20	30	133	6	56	212	3845
	(c)	20	50	115	440	14	170	1073	19452
	(d)	10	10	20	30	8	27	127	753
	(e)	10	20	20	40	8	53	140	1067
	(f)	10	20	40	110	6	53	260	2780
	(g)	10	20	30	40	7	50	182	939
	(h)	10	30	60	190	6	80	392	4828
	(i)	10	25	55	130	7	67	382	3823
	(j)	10	20	110	140	5	30	704	6647
	(k)	10	20	35	115	7	53	225	2929
	(l)	10	30	125	180	5	75	1092	8220
	(m)	10	20	40	105	15	54	257	2730
	(n)	20	85	165	370	14	275	1478	15619
	(o)	25	30	60	175	19	99	486	7549

Table 4.12: Median thinning values Δ for the 20 runs of the sampler corresponding to the 20 different random samples for 4 different samples of the fifteen mixtures of normals reported in Table 4.1. Also, the allocation sampler median run times are reported for each mixture in seconds.

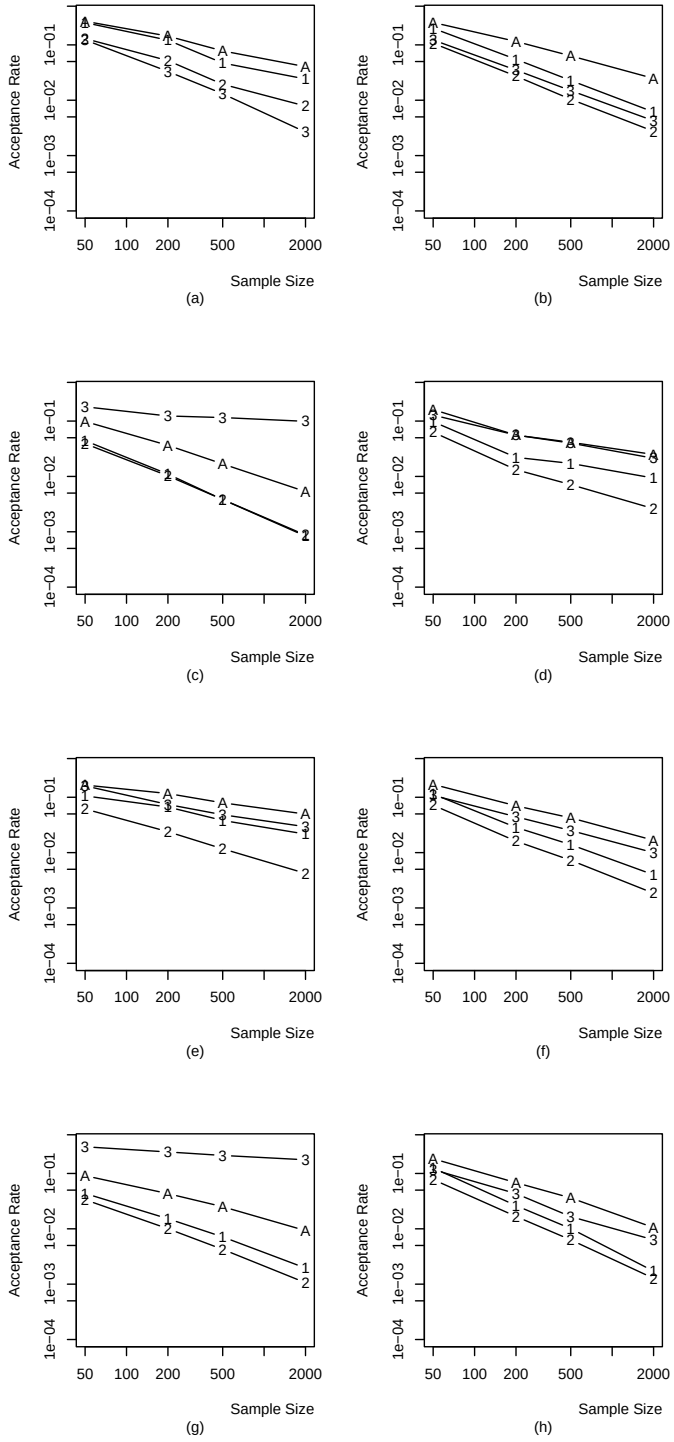

Figure 4.17: Plots of acceptance rates of the Absorption/Ejection move (labelled A) and the three Metropolis-Hastings moves (labelled 1, 2, 3) against sample size, on a doubly logarithmic scale. Each plot shows the acceptance rates averaged over the 20 runs of the allocation sampler for the mixture detailed in Table 4.1.

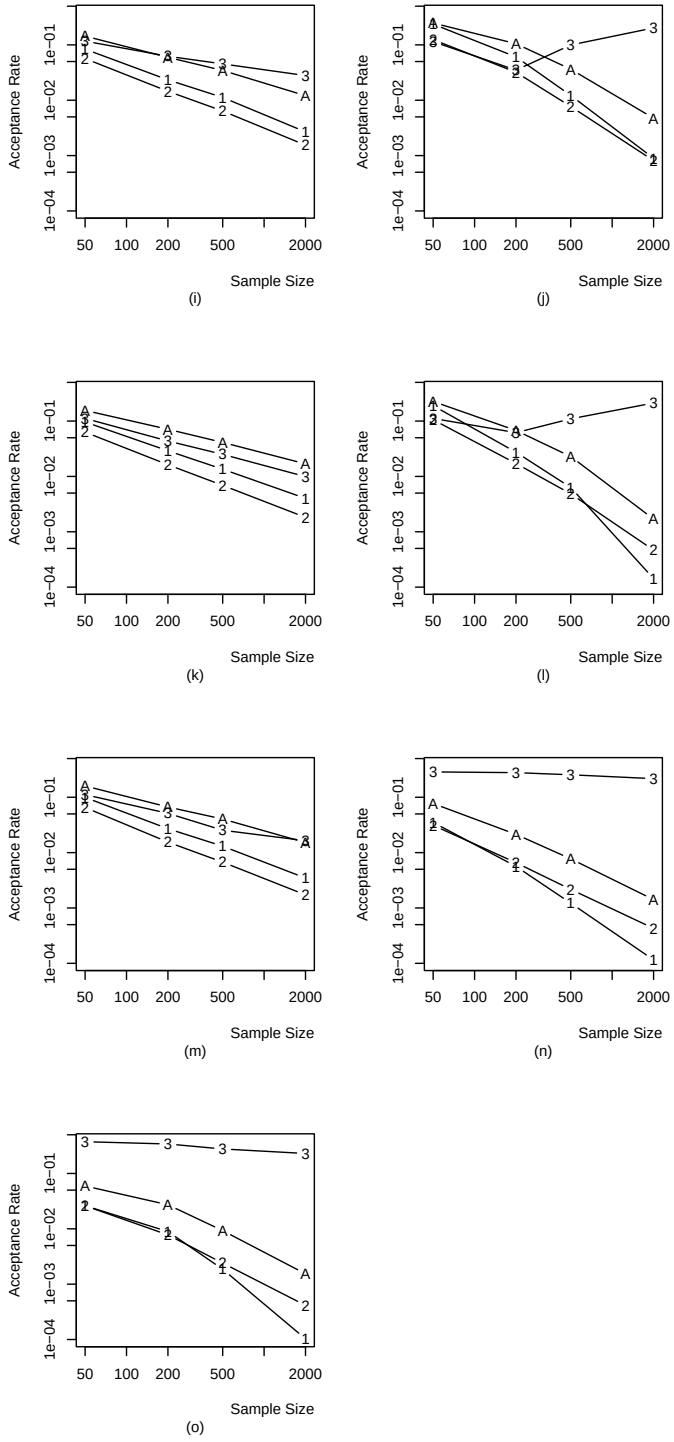

Figure 4.18: Plots of acceptance rates of the Absorption/Ejection move (labelled A) and the three Metropolis-Hastings moves (labelled 1, 2, 3) against sample size, on a doubly logarithmic scale. Each plot shows the acceptance rates averaged over the 20 runs of the allocation sampler for the mixture detailed in Table 4.1.

Chapter 5

Real Dataset Examples

In this chapter a selection of different datasets will be analysed using the allocation sampler. The results will be documented and compared to previous analyses carried out by other authors. For each dataset a preliminary run was executed to fix the hyperparameter and thinning values for five further runs of the sampler. All the other settings for the allocation sampler were chosen in exactly the same way as described for the simulation study in Chapter 4.

5.1 Galaxy

This dataset contains the velocities, in 10^3 km/sec, of 82 distant galaxies in the Corona Borealis region of the universe that are diverging from our own galaxy. These velocities are proportional to the distance between our galaxy and these other 82 galaxies. A histogram of the data is shown in Figure 5.1. Astronomers have postulated about the presence of clusters of galaxies and thus for this to be true the velocities should follow a multimodal distribution, with every mode

corresponding to a different cluster. The original dataset, analysed in Postman et al. (1986), actually had 83 data observations but an observation was dropped in the analysis by Roeder (1990). It is these 82 observations that have been analysed using mixture models by numerous other authors including Richardson and Green (1997), Stephens (2000a), Nobile (1994) and McGrory (2005). This dataset was modelled using a mixture of normals and also a mixture of uniforms.

The following parameter settings were set after a preliminary run of the allocation sampler when fitting mixtures of normal components and uniform components:

$$\text{Normal}: \ \mu = 20.83,\ \tau = 0.04,\ \gamma = 2,\ \delta = 2,\ \Delta = 70,$$
$$\text{Uniform}: \ \phi = 40,\ \Delta = 60.$$

The allocation sampler favours a mixture of between 4 and 6 normal components, see Table 5.1 for the posterior of k, with the highest posterior probability being assigned to a mixture of 5 components. The RJMCMC method of Richardson and Green (1997) preferred a mixture of between 5 and 7 normal components with 6 having the highest probability. However, the analyses in Stephens (2000a) and McGrory (2005) both selected a normal mixture of 3 components. Furthermore, the likelihood ratio test approach to assessing the number of components from MacLachlan (1997) selected 6 components. Therefore, the results of the allocation sampler are not dissimilar to the results of these previous analyses. A summary of five previous Bayesian analyses of this dataset is given in Aitkin (2001) that give evidence for models between 3 and 9 components. The posterior predictive density is shown in Figure 5.1 superimposed on a histogram of the data. This dataset was also analysed using a mixture of uniform components. The posterior

of k is shown in Table 5.2 with the high posterior probabilities showing up for mixtures of 2 and 3 components. The posterior predictive density is shown in Figure 5.1. Comparing this distribution to that for the mixture of normals it is of no surprise that the mixture of uniforms posterior predictive density is much less smooth. Also, Table 5.3 reports the estimates of the parameters for the 5-component mixture. This table shows that the middle section of the data from the values around 15 to 28 is modelled by 3 components.

k	1	2	3	4	**5**	6	7	8	9	10
$\pi(k\|x)$	0.000	0.000	0.090	0.289	**0.337**	0.200	0.067	0.015	0.002	0.000
s.d.	0.000	0.000	0.009	0.006	**0.007**	0.008	0.003	0.001	0.001	0.000

Table 5.1: Posterior distribution of k for the galaxy dataset using univariate normal components. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

k	1	**2**	3	4	5	6	7
$\pi(k\|x)$	0.000	**0.527**	0.402	0.065	0.005	0.000	0.000
s.d.	0.000	**0.023**	0.018	0.005	0.001	0.000	0.000

Table 5.2: Posterior distribution of k for the galaxy dataset using uniform components. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

Component	1	2	3	4	5
Weights	0.09	0.10	0.32	0.43	0.06
Means	9.78	19.75	19.88	22.64	31.86
St. Dev.	1.05	1.39	0.83	1.69	1.20

Table 5.3: Estimates of the parameters in the 5-component mixture of normals for the galaxy dataset after post-processing.

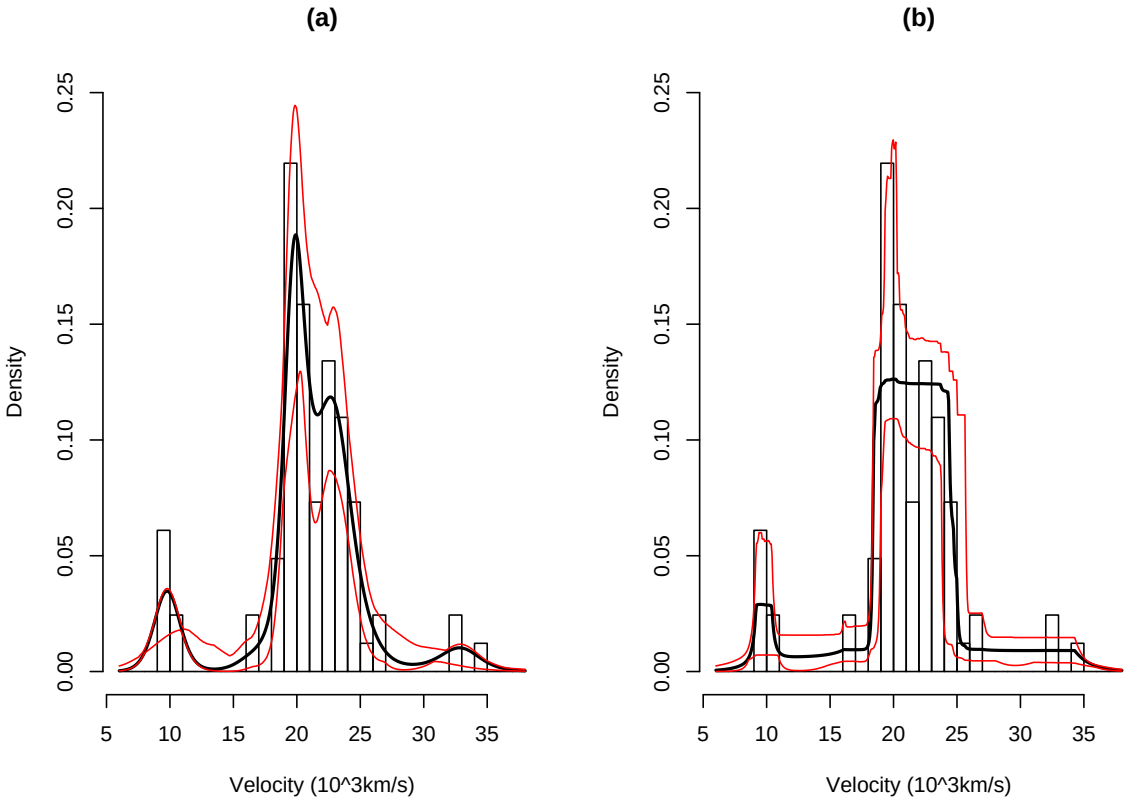

Figure 5.1: Histograms and posterior predictive densities for the galaxy dataset. Graph (a) corresponds to a mixture of normals and graph (b) to a mixture of uniforms. The black line is the estimate of the posterior predictive density and the red lines show the 0.005 and 0.995 quantiles of the simulated densities.

5.2 Acidity

This dataset consists of acid neutralising capacity index scores for a sample of 155 lakes in North-Central Wisconsin. These data were log-transformed and analysed using a mixture of normal distributions by Crawford et al. (1992). The dataset analysed here is also on a log-scale. Mixtures of normals were also fitted on this log-scale dataset by Richardson and Green (1997) and McGrory (2005).

A preliminary run of the allocation sampler gave rise to the following parameter values for the further 5 runs:

$$\mu = 5.11,\ \tau = 0.2,\ \gamma = 2,\ \delta = 0.27,\ \Delta = 30.$$

The posterior distribution of k reported in Table 5.4 shows that the models with between 2 and 4 components account for approximately 93% of the distribution. The 3-component model has the highest posterior probability. This result is similar to the analysis of Richardson and Green (1997). They found support for models of between 3 and 5 components, with the highest probability given to the model of 3 components. Another analysis by McGrory (2005) using the variational Bayes technique fitted a 2-component model which again seems reasonable when comparing the results found by the allocation sampler. Also, the likelihood approach from MacLachlan (1997) found 3 components. Again, the allocation sampler produces results that compare well to previous analyses. The parameter estimates for the 3-component model can be found in Table 5.5. Comparing these to the 2-component model of McGrory (2005) it appears that components 1 and 2 here are joined together to make the first component in McGrory (2005).

k	1	2	**3**	4	5	6	7	8
$\pi(k\|x)$	0.000	0.209	**0.476**	0.236	0.064	0.013	0.002	0.000
s.d.	0.000	0.008	**0.008**	0.007	0.003	0.001	0.000	0.000

Table 5.4: Posterior distribution of k for the acidity dataset. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

Component	1	2	3
Weights	0.45	0.21	0.34
Means	4.28	4.91	6.35
St. Dev.	0.31	0.58	0.46

Table 5.5: Estimates of the parameters in the 3-component mixture of normals for the acidity dataset after post-processing.

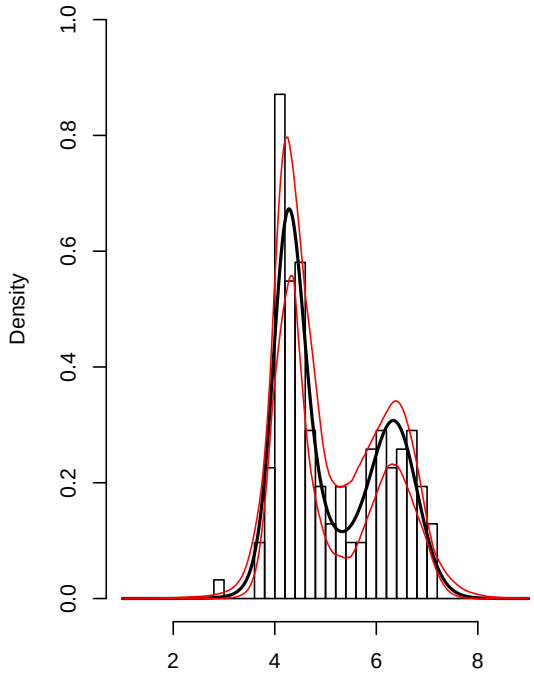

Figure 5.2: Histogram and posterior predictive density for the acidity dataset. The black line is the estimate of the posterior predictive density and the red lines show the 0.005 and 0.995 quantiles of the simulated densities.

5.3 Enzyme

This dataset arises from interest in enzymatic activity in the blood, for an enzyme involved in the metabolism of the carcinogenic substances in order to identify subgroups of slow or fast metabolisers. These subgroups can then be proposed as markers of genetic polymorphism in the wider population. The dataset contains measurements for a group of 245 unrelated individuals. Bechtel et al. (1993) analysed the data by fitting a mixture of two skewed distributions using maximum likelihood techniques.

The following parameter values were chosen from the usual preliminary run of the allocation sampler:

$$\mu = 0.62,\ \tau = 0.069,\ \gamma = 2,\ \delta = 0.024,\ \Delta = 70.$$

From Table 5.6 one can see that the allocation is favouring a model with $3-4$ components with the highest posterior probability on the 3-component model. This agrees with the previous analyses of other authors. The methods of Richardson and Green (1997) and McGrory (2005) both picked out a 4-component model. The analysis by Richardson and Green (1997) assigned very similar posterior probabilities to the 3-component and 4-component models, with a difference of only 0.027. Comparing Table 5.7 to the component parameters calculated in McGrory (2005), the difference between the 3-component and 4-component models seems to be due to the first component being either grouped as one or split into two separate components.

k	1	2	**3**	4	5	6	7	8
$\pi(k\|x)$	0.000	0.049	**0.486**	0.343	0.102	0.018	0.002	0.000
s.d.	0.000	0.005	**0.018**	0.011	0.004	0.001	0.000	0.000

Table 5.6: Posterior distribution of k for the enzyme dataset. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

Component	1	2	3
Weights	0.60	0.15	0.25
Means	0.19	1.08	1.46
St. Dev.	0.08	0.17	0.50

Table 5.7: Estimates of the parameters in the 3-component mixture of normals for the enzyme dataset after post-processing.

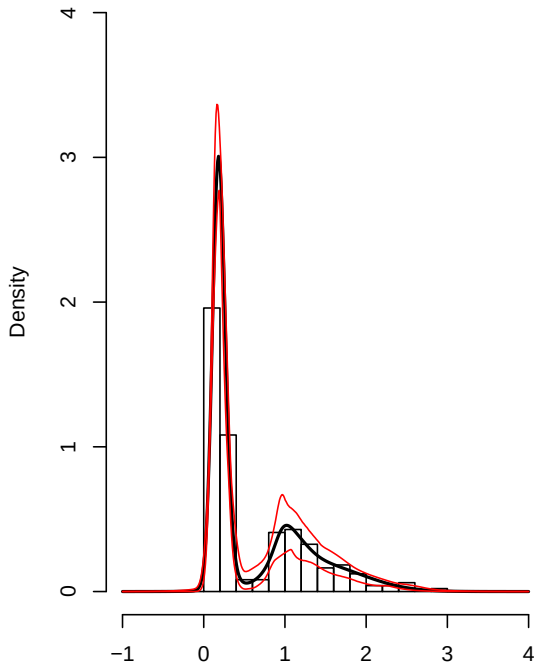

Figure 5.3: Histogram and posterior predictive density for the enzyme dataset. The black line is the estimate of the posterior predictive density and the red lines show the 0.005 and 0.995 quantiles of the simulated densities.

5.4 Hidalgo Stamps

This is a dataset concerned with the 1872 Hidalgo stamp issue in Mexico. The dataset consists of the thicknesses, in (mm), of 485 unwatermarked, used, white wove stamps. Furthermore, 289 had an 1872 overprint and 196 of the stamps had an overprint from either 1873 or 1874. A more detailed description of the dataset can found in Izenman and Sommer (1988).

From a preliminary run of the allocation sampler the following parameters were selected and fixed for the 5 further runs:

$$\mu = 0.086,\ \tau = 0.027,\ \gamma = 2,\ \delta = 0.000008,\ \Delta = 118.$$

The analysis in Izenman and Sommer (1988) used a likelihood approach to fit a finite mixture model. They reported that they found a 7-component normal mixture using a non-parametric approach, but, by applying the test from Wolfe (1971) as a parametric alternative, they found that a 3-component mixture should be fitted. This conflict was further analysed by Basford et al. (1997) and they argued that using a homoscedastic model is appropriate for the parametric and non-parametric approaches of Izenman and Sommer (1988) to coincide. A Bayesian approach using a Bayes factor method for finding the number of components was taken by Ishwaran et al. (2001). These authors reported that an 8-component model describes the data in the best possible manner. The allocation sampler produced slightly different results from these methods. Table 5.8 shows the posterior of k giving significant support to a model of between 3 and 5 components. The 4 component model has the highest posterior probability but note that the posterior of k assigns a probability of more than 2% to models of

between 3 and 9 components which is a large spread of plausible values. The parameter estimates for the 4 component model are reported in Table 5.9.

k	1	2	3	**4**	5	6
$\pi(k\|x)$	0.000	0.000	0.181	**0.449**	0.162	0.059
s.d.	0.000	0.000	0.009	**0.006**	0.007	0.008
k	7	8	9	10	11	12
$\pi(k\|x)$	0.062	0.054	0.024	0.007	0.001	0.000
s.d.	0.003	0.001	0.001	0.000	0.000	0.000

Table 5.8: Posterior distribution of k for the stamps dataset. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

Component	1	2	3	4
Weights	0.18	0.13	0.26	0.43
Means	0.071	0.077	0.080	0.099
St. Dev.	0.001	0.002	0.002	0.014

Table 5.9: Estimates of the parameters in the 4-component mixture of normals for the Hidalgo stamps dataset after post-processing.

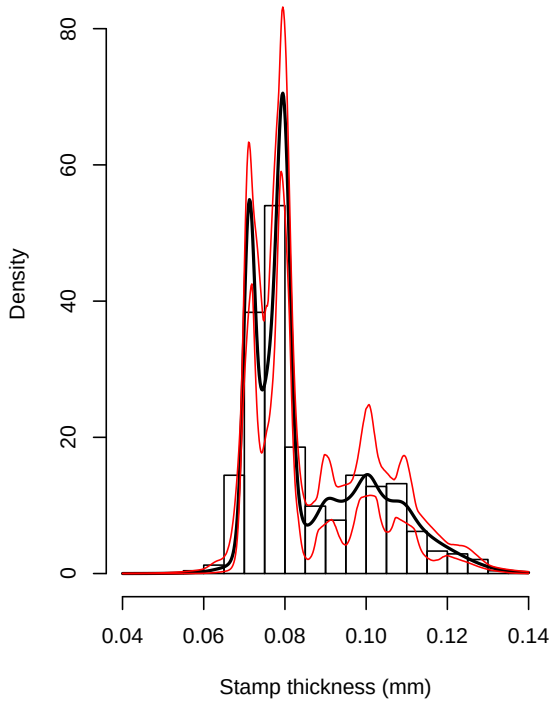

Figure 5.4: Histogram and posterior predictive density for the Hidalgo stamps dataset. The black line is the estimate of the posterior predictive density and the red lines show the 0.005 and 0.995 quantiles of the simulated densities.

5.5 S&P 500 Returns

As an example of density estimation using a mixture of sign-shifted exponential distributions a dataset from the Standard and Poors 500 (S&P 500) stock index is considered. This dataset consists of 1700 observations of daily returns from the 1950s and was previously analysed by Rydén et al. (1998) using a hidden Markov model where it is referred to as subseries E. Robert et al. (2000) also analysed this dataset using hidden Markov models with zero-mean normal distributions by implementing a reversible jump sampler. Their analysis found that there were 2 or 3 components in the data. Here, any possible dependence between the observations in the dataset is not accounted for.

The preliminary run of the allocation sampler gave rise to the following settings for the next 5 runs:

$$\kappa = 1,\ \beta = 0.004,\ \gamma = 10,\ \rho = 0.5,\ \Delta = 1000.$$

Looking at a plot of the data, see Figure 5.6, these hyperparameter values seem reasonable. The γ value is of particular interest in that the fairly large value of 10 restricts the shift parameter to be very close to zero. The allocation sampler found that a 3-component mixture is best for modelling this S&P 500 data, see Table 5.10. Table 5.11 displays estimates of the mixture parameters for each component. The Table shows that there are 2 components that decay in a positive direction but only 1 component that decays in the negative direction. It would be sensible to comment that the 2 components are required for the positive direction so as to model the tails of the data. Also, the marginal posterior density estimates for the shift and rate parameters, conditional on the sign parameter, are shown in Figure

5.5. Inspection of these plots along with the parameter estimates given in Table 5.11 show that all the shift parameters are very close to zero as would be expected. However, the rate parameters seem to be fairly similar for all the components, implying a fairly stable rate of decay in both directions. The posterior predictive density shown in Figure 5.6 has one unusual feature in the model due to the shift parameters not quite being equal. If one fixes the model to have the shift parameter exactly equal to zero then a 2-component model is selected by the allocation sampler, one component for each direction of decay.

k	1	2	**3**	4	5	6	7
$\pi(k\|x)$	0.000	0.006	**0.817**	0.157	0.019	0.002	0.000
s.d.	0.000	0.001	**0.023**	0.019	0.003	0.001	0.000

Table 5.10: Posterior distribution of k for the S&P 500 returns dataset. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

Component	1	2	3
Weights	0.47	0.33	0.20
Signs	-1	1	1
Rates	201	215	210
Shifts	-0.0003	-0.0004	0.0003

Table 5.11: Estimates of the parameters in the 3-component mixture of sign-shifted exponentials for the S&P 500 returns dataset after post-processing.

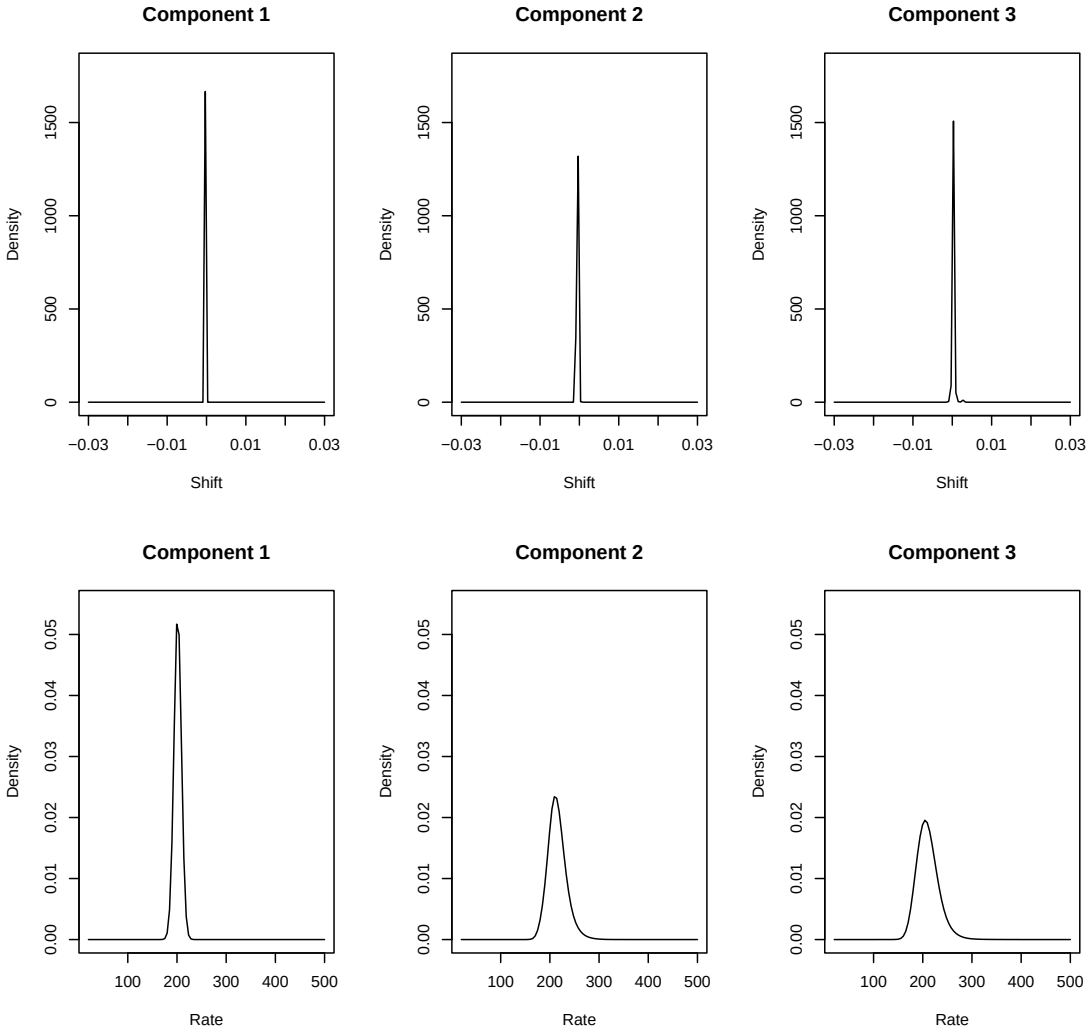

Figure 5.5: Marginal posterior density estimates for the shift and rate parameters, conditional on the sign parameter given in Table (5.11), in the 3-component mixture of sign-shifted exponential distributions for the S&P 500 Returns dataset

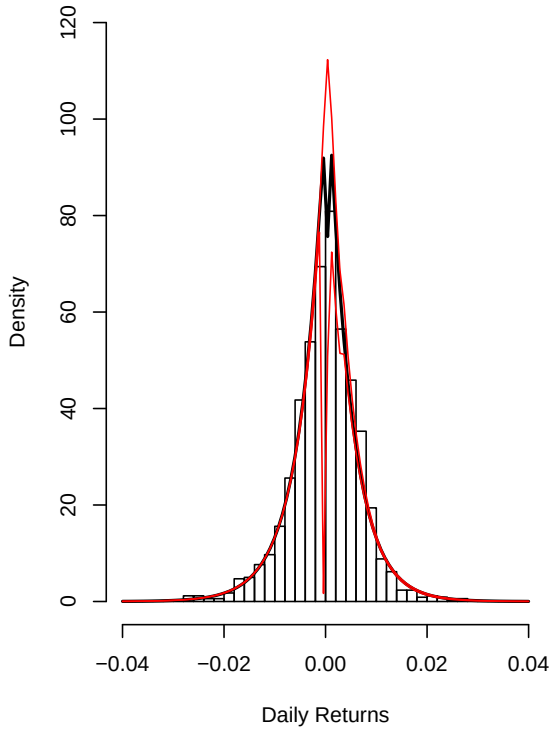

Figure 5.6: Histogram and posterior predictive density for the S&P 500 Returns dataset. The black line is the estimate of the posterior predictive density for fixed $k = 3$ and the red lines show the 0.005 and 0.995 quantiles of the simulated densities.

5.6 Iris

This is the famous dataset sometimes referred to as the "Fisher Iris Data" even though it was actually collected by Dr. Edgar Anderson. The data were published in Fisher (1936) and contain 4 measurements, namely sepal length and width and petal length and width, in centimetres for 50 plants for each of three different species of iris: Iris setosa, Iris versicolor and Iris virginica. The author used the data to illustrate the use of discriminant functions. The allocation sampler was implemented using the data on the measurements to find the number of species. Therefore, knowledge about the species was not used in the fitting of the model. This dataset also allows an illustration of the clustering capabilities of the allocation sampler.

The following parameter settings were used for 5 runs of the allocation sampler:

$$\xi = \begin{bmatrix} 0.55 & 0 & 0 & 0 \\ 0 & 0.40 & 0 & 0 \\ 0 & 0 & 0.35 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \mu = \begin{bmatrix} 5.84 \\ 3.06 \\ 3.76 \\ 1.20 \end{bmatrix}, \tau = 0.065, \eta = 7, \Delta = 360.$$

It is reassuring to see from Table 5.12 that the allocation sampler selects a 3-component mixture model, to correspond to the 3 different species of iris, with a posterior probability of 0.72. There is also a significant amount of mass, 0.27, on a 4-component mixture model. All the univariate and bivariate marginal posterior predictive distributions are displayed in Figure 5.7. The bivariate plots show that the Setosa species is easily separated from the Versicolor and Virginica

species. Furthermore, they show that there is some overlap between the Versicolor and Virginica species. The univariate marginal distributions show that the petal width variable is possibly the best predictor of species, due to its trimodality, and sepal width the worst because of the unimodal nature of that posterior predictive density. Figure 5.8 is displayed to show the ability of the allocation sampler to separate the observations into clusters. This Figure is an image plot where the value at the $(i, j)^{th}$ position is found using formula (2.44). It shows that all the Setosa observations are always correctly grouped together. Also, the allocation sampler groups the majority of Versicolor observations into the same component, but, there are 5 observations that are sometimes placed in a group with Virginica observations. Finally, only 1 Virginica observation seems to be occasionally incorrectly assigned to the Versicolor group. These results compare well to analyses by previous clustering methods such as the k-means clustering algorithm (MacQueen (1967)), Ward's algorithm (Ward (1963)) and the cluster-function-based method (Li (2006)). Li (2006) presents results for all three of these methods for this dataset. The results show that each method always classifies the Setosa observations correctly but misclassfies some of the Virginica and Versicolor observations in a similar fashion to the allocation sampler.

k	1	2	**3**	4	5	6
$\pi(k\|x)$	0.000	0.002	**0.718**	0.267	0.013	0.000
s.d.	0.000	0.004	**0.009**	0.008	0.001	0.000

Table 5.12: Posterior distribution of k for the iris dataset. The probability $\pi(k|x)$ is the average of five estimates from independent runs of the allocation sampler; s.d. is the standard deviation of the five estimates.

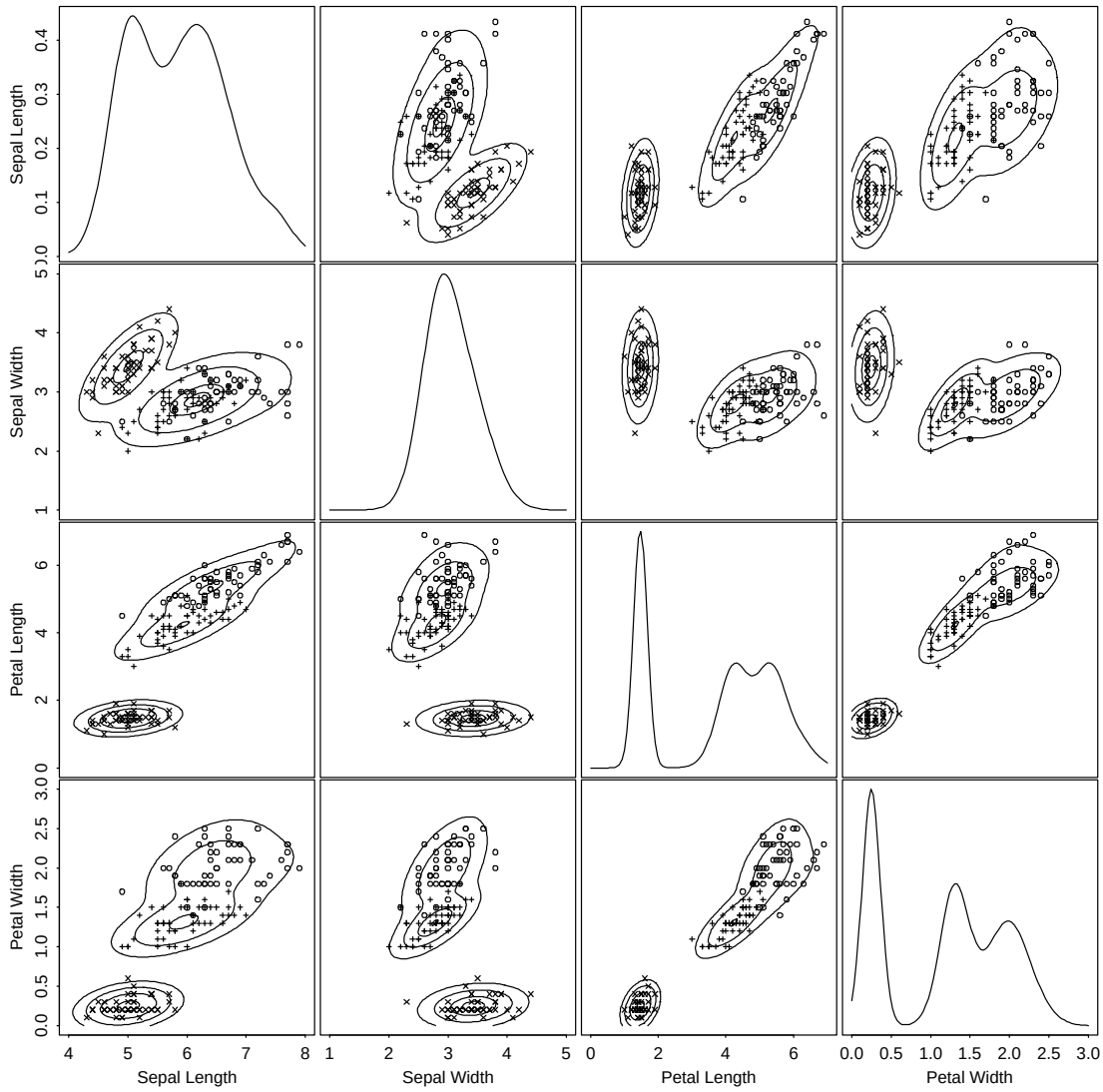

Figure 5.7: Posterior predictive distribution for the iris dataset. Univariate marginal densities, located on the diagonal, and the bivariate marginal densities, on the off-diagonal, of an estimate of the four-dimensional posterior predictive density. In the bivariate plots, contour lines are drawn at levels corresponding to 5%, 25%, 75% and 95% of the posterior probability of the displayed region. Overlaid on the contour plots are bivariate scatterplots of the data, using the symbols x = Setosa, + = Versicolor, o = Virginica.

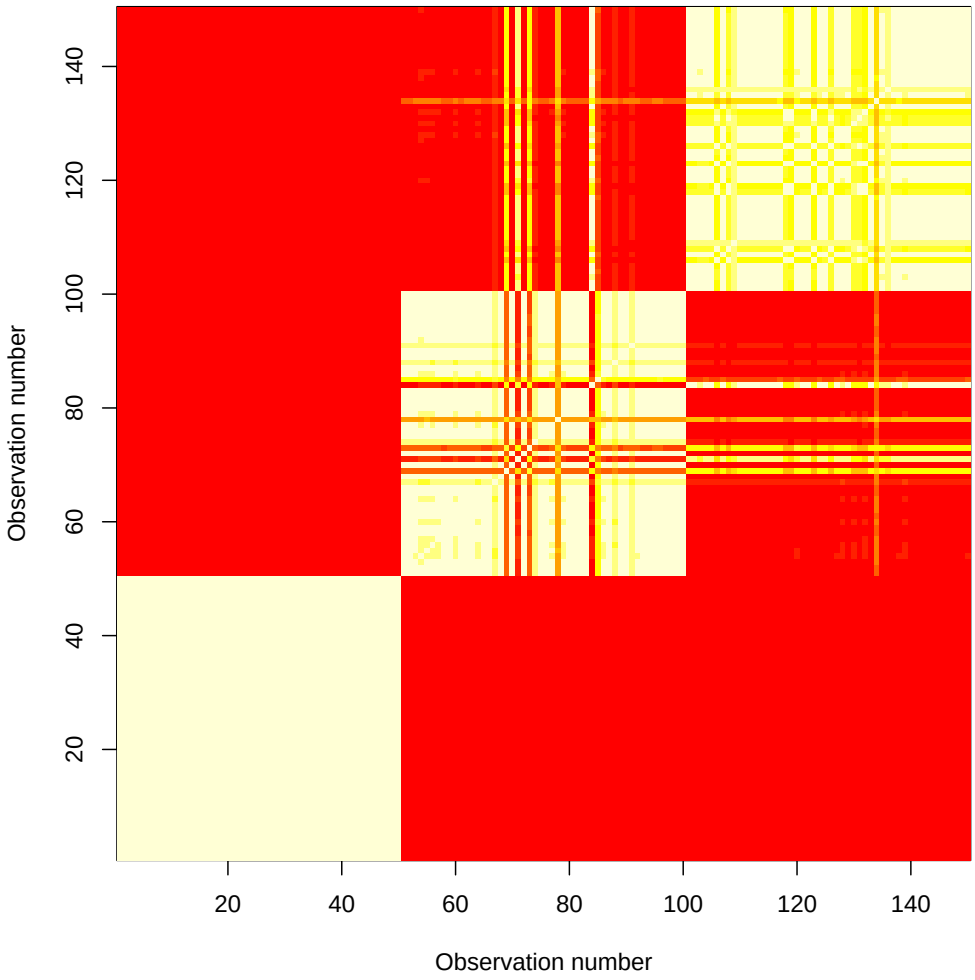

Figure 5.8: Image plot of pairwise classification probabilities for the iris dataset. The observations are numbered so that the observations 1 to 50 are from the Setosa species, 51 to 100 from the Versicolor species and 101 to 150 from the Virginica species. The cream colour corresponds to a probability of 1, the red colour to a probability of 0 and the various shades of yellow to values between 0 and 1.

Chapter 6

Conclusions & Future Research

This chapter will state, summarise and discuss the conclusions that can be drawn from the work presented in this thesis. Also, some possible extensions of this work will be considered.

Chapter 1 reviewed the problem and theory of fitting a finite mixture distribution. It was seen that there have been many attempts by numerous authors to analyse these distributions in order to make use of their flexible properties. The problem of finding the number of components required to fit the data has been tackled in many different ways but a definitive solution has not been found. Current methods available to attack this problem, include, among others, RJMCMC and Birth-Death processes. The chapter also brought up the other problems that arise when analysing mixture models, e.g. the label switching problem. This led into Chapter 2 where a Bayesian model was defined, that is suitable for the new approach of the allocation sampler. This Bayesian model makes the assumption that the component parameters and weights can be integrated out of the model

analytically in closed form. This assumption means that the state space for the MCMC sampler is simply the set of all allocation vectors g, where g contains the component to which each observation has been allocated, and the number of these components k is allowed to vary. This is a reduction in the size and complexity of the state space when compared to the model used by Richardson and Green (1997) and by Stephens (2000a), which also includes the component parameters and weights. Component parameters and weights can be integrated out in the case where the component distributions are from an exponential family with a conjugate prior on the parameters,see Sections 2.4.1 and 2.4.2. Moreover, closed form integration is still possible for some other distributions which do not admit conjugate priors, examples were provided in Sections 2.4.3 and 2.4.4.

A method for analysing the Bayesian model defined in Chapter 2 was then described in Chapter 3. This model required the invention of a MCMC sampler that could not not only move within models but also between models in a reversible jump type manner. This was achieved with the definition of four within model moves and one between model move. A big advantage of this algorithm is that the moves do not depend on the distribution of the components because the state space is always the same no matter the distributional form of the components. Therefore, implementing the sampler for other distributions requires only minimal changes if compared to standard RJMCMC. The design stage of the allocation sampler brought forward further problems that had to be solved, such as how to set hyperparameter values for the prior distributions and how to counteract the problem of label-switching. All the posterior analysis presented in this thesis is conditional on these hyperparameters so the selection

of their values play a crucial role in the effectiveness of the allocation sampler. The procedure used to select the hyperparameters produced satisfactory results, however, trying to make it a less subjective method in the future would enhance this approach. The label-switching problem had to be overcome to enable meaningful parametric inference. A Metropolis-Hastings move on the labels was used in the case where the prior distributions of the components were asymmetric. However, a post-processing algorithm was implemented for the symmetric prior case. The post-processing algorithm was shown in Chapter 4 to yield very good results in removing from the posterior distributions of the parameters the symmetry, typically induced by label-switching. Furthermore, the parameter estimates calculated after the removal of the symmetry were extremely close to the true values, where they were known. Within Chapters 4 and 5 the allocation sampler was shown to obtain good posterior fits to the datasets whether they were simulated or real in the majority of cases. The important problem of selecting the true number of components, also gave very promising results. This was again illustrated using both simulated and real data in Chapters 4 and 5.

In this thesis an example was given showing how the finite mixture model can be used as a cluster analysis tool. Estimates of the posterior probabilities that two observations are allocated to the same component were shown for the Iris dataset in Figure 5.8. Further work could be done on looking at the possibility of turning the allocation sampler's output into a tool to perform meaningful cluster analysis and a method for creating discriminant rules. This would require a more in depth analysis of the posterior distribution of g.

There are some other possible areas in which the allocation sampler could

progress. Firstly, a referee of Nobile and Fearnside (2007) suggested that this framework might be applicable to the admixture model proposed in Pritchard et al. (2000). This paper concerns genotype data and tries to ascertain a population structure and assign individuals to the populations. An obvious way of improving the capabilities of the allocation sampler would be to include more and more possible distributions for the components to take. In addition, another extension could be to allow the distribution of a component to change throughout the sampling process to enable mixtures of more than one type of distribution, e.g. mixtures of normals and exponentials. To implement this a Metropolis-Hastings move could be designed such that it proposes a change to the distribution of the component. A RJMCMC sampler for hidden Markov models is defined in Robert et al. (2000) where they try to find the number of components in the model and estimate the parameters. Therefore, another way of enhancing the allocation sampler would be to allow dependence between the observations therefore using a hidden Markov model structure. However, this adaption of the model may require the addition of new moves to the allocation sampler.

Appendix A

Integrating parameters from the model

This appendix contains the details of the integral

$$p(x|\phi) = \int \prod_i q(x_i|\theta)\pi(\theta|\phi)\mathrm{d}\theta$$

which defines the marginal distribution of the data x where the prior on the parameters $\pi(\theta|\phi)$ is non-conjugate.

A.1 Uniform Distribution

Let $x = (x_1, \ldots, x_n)$ be i.i.d. $\sim Unif(a, b)$. Then

$$q(x|\theta) = \frac{1}{(b-a)^n} I_{(-\infty, x_{(1)})}(a) I_{(x_{(n)}, \infty)}(b),$$

where $\theta = (a, b)$ and I is the indicator function.

Then

$$p(x|\phi) = \int q(x|\theta)\pi(\theta|\phi)\mathrm{d}\theta.$$

Suppose the prior on the parameters (a, b) is defined as

$$\pi(a, b) = \tfrac{1}{2\phi^2}, \quad -\phi < a < b < \phi,$$

where ϕ is a known hyperparameter.

The marginal distribution of x, $p(x|\phi)$, can be calculated in closed form as follows:

$$\begin{aligned} p(x|\phi) &= \int\int q(x|a,b)\pi(a,b)\mathrm{d}a\mathrm{d}b \\ &= \int\int \frac{1}{2\phi^2}\frac{1}{(b-a)^n} I_{(-\infty,x_{(1)})}(a) I_{(x_{(n)},\infty)}(b)\mathrm{d}a\mathrm{d}b, \quad -\phi < a < b < \phi. \end{aligned}$$

A change of variable is required to progress, so let $z = a$ and $y = b - a$. Thus $p(x|\phi)$ becomes

$$\begin{aligned} p(x|\phi) &= \int\int \frac{1}{2\phi^2}\frac{1}{y^n} I_{(-\infty,x_{(1)})}(z) I_{(x_{(n)},\infty)}(y+z)\mathrm{d}z\mathrm{d}y, \quad -\phi < z < (y+z) < \phi \\ &= \frac{1}{2\phi^2}\int\int \frac{1}{y^n} I_{(-\infty,x_{(1)})}(z) I_{(-\phi,\phi)}(z) I_{(x_{(n)}-z,\infty)}(y) I_{(0,\phi-z)}(y)\mathrm{d}z\mathrm{d}y \\ &= \frac{1}{2\phi^2}\int\int \frac{1}{y^n} I_{(-\phi,x_{(1)})}(z) I_{(x_{(n)}-z,\phi-z)}(y)\mathrm{d}z\mathrm{d}y \\ &= \frac{1}{2\phi^2}\int I_{(-\phi,x_{(1)})}(z)\int_{x_{(n)}-z}^{\phi-z}\frac{1}{y^n}\mathrm{d}y\mathrm{d}z. \end{aligned} \tag{A.1}$$

The integral with respect to y has two different cases which need to be examined

separately. Firstly, for the case $n = 1$ note that $x_{(n)} = x_{(1)}$ and then

$$\begin{aligned}
p(x|\phi) &= \frac{1}{2\phi^2}\int I_{(-\phi,x_{(1)})}(z)\int_{x_{(1)}-z}^{\phi-z}\frac{1}{y}\mathrm{d}y\mathrm{d}z \\
&= \frac{1}{2\phi^2}\int I_{(-\phi,x_{(1)})}(z)\left[\log(y)\right]_{x_{(1)}-z}^{\phi-z}\mathrm{d}z \\
&= \frac{1}{2\phi^2}\int_{-\phi}^{x_{(1)}}\left[\log(\phi-z)-\log(x_{(1)}-z)\right]\mathrm{d}z \\
&= \frac{1}{2\phi^2}\left(\int_{-\phi}^{x_{(1)}}\log(\phi-z)\mathrm{d}z-\int_{-\phi}^{x_{(1)}}\log(x_{(1)}-z)\mathrm{d}z\right) \qquad \text{(A.2)} \\
&= \frac{1}{2\phi^2}\Bigg(\Big[(\phi-z)\log(\phi-z)-(\phi-z)\Big]_{x_{(1)}}^{-\phi} \\
&\qquad -\Big[(x_{(1)}-z)\log(x_{(1)}-z)-(x_{(1)}-z)\Big]_{x_{(1)}}^{-\phi}\Bigg) \qquad \text{(A.3)} \\
&= \frac{1}{2\phi^2}\Big(2\phi\log[2\phi]-(\phi-x_{(1)})\log[\phi-x_{(1)}]-(\phi+x_{(1)})\log[\phi+x_{(1)}]\Big) \\
&= \frac{1}{2\phi^2}\left(x_{(1)}\log\left[\frac{\phi-x_{(1)}}{\phi+x_{(1)}}\right]+\phi\log\left[\frac{(2\phi)^2}{(\phi+x_{(1)})(\phi-x_{(1)})}\right]\right). \qquad \text{(A.4)}
\end{aligned}$$

Note that in relation to (A.2) the indefinite integral of $\log(c-x)$ with respect to x, where c is a constant, can be defined as $-\{(c-x)\log(c-x)-(c-x)\}$. Consequently, the limits swap order in (A.3).

Returning to the integral (A.1) for the case when $n > 1$, we have

$$\begin{aligned}
p(x|\phi) &= \frac{1}{2\phi^2}\int I_{(-\phi,x_{(1)})}(z)\left[-\frac{1}{n-1}\frac{1}{y^{n-1}}\right]_{x_{(n)}-z}^{\phi-z}\mathrm{d}z \\
&= \frac{1}{2\phi^2}\int I_{(-\phi,x_{(1)})}(z)\left[\frac{1}{n-1}\left(\frac{1}{(x_{(n)}-z)^{n-1}}-\frac{1}{(\phi-z)^{n-1}}\right)\right]\mathrm{d}z \\
&= \frac{1}{2\phi^2}\frac{1}{n-1}\left[\int_{-\phi}^{x_{(1)}}\frac{1}{(x_{(n)}-z)^{n-1}}\mathrm{d}z-\int_{-\phi}^{x_{(1)}}\frac{1}{(\phi-z)^{n-1}}\mathrm{d}z\right]. \qquad \text{(A.5)}
\end{aligned}$$

Again, these integrals also have two cases, $n = 2$ and $n > 2$. Firstly, for $n = 2$,

$$\begin{aligned} p(x|\phi) &= \frac{1}{2\phi^2}\left(\int_{-\phi}^{x_{(1)}}(x_{(2)}-z)^{-1}\mathrm{d}z - \int_{-\phi}^{x_{(1)}}(\phi-z)^{-1}\mathrm{d}z\right) \\ &= \frac{1}{2\phi^2}\Bigg(\log(x_{(2)}+\phi) - \log(x_{(2)}-x_{(1)}) \\ &\qquad + \log(\phi - x_{(1)}) - \log(2\phi)\Bigg) \\ &= \frac{1}{2\phi^2}\log\left[\frac{(x_{(2)}+\phi)(\phi-x_{(1)})}{(x_{(2)}-x_{(1)})2\phi}\right]. \end{aligned} \tag{A.6}$$

Next, for the case $n > 2$ (A.5) becomes

$$\begin{aligned} p(x|\phi) &= \frac{1}{2\phi^2}\frac{1}{n-1}\left(\int_{-\phi}^{x_{(1)}}(x_{(n)}-z)^{-n+1}\mathrm{d}z - \int_{-\phi}^{x_{(1)}}(\phi-z)^{-n+1}\mathrm{d}z\right) \\ &= \frac{1}{2\phi^2}\frac{1}{n-1}\Bigg(\frac{-1}{n-2}\big[(x_{(n)}+\phi)^{-n+2} - (x_{(n)}-x_{(1)})^{-n+2} \\ &\qquad + (\phi - x_{(1)})^{-n+2} - (2\phi)^{-n+2}\big]\Bigg) \\ &= \frac{1}{2\phi^2(n-1)(n-2)}\Bigg((x_{(n)}-x_{(1)})^{-n+2} - (x_{(n)}+\phi)^{-n+2} \\ &\qquad - (\phi - x_{(1)})^{-n+2} + (2\phi)^{-n+2}\Bigg) \\ &= \frac{(x_{(n)}-x_{(1)})^{-n+2} - (x_{(n)}+\phi)^{-n+2} - (\phi-x_{(1)})^{-n+2} + (2\phi)^{-n+2}}{2\phi^2(n-1)(n-2)}. \end{aligned} \tag{A.7}$$

Thus, collating (A.4), (A.6) and (A.7) produces

$$p(x|\phi) = \begin{cases} \frac{1}{2\phi^2}\frac{\left[(x_{(n)}-x_{(1)})^{-n+2}-(x_{(n)}+\phi)^{-n+2}-(\phi-x_{(1)})^{-n+2}+(2\phi)^{-n+2}\right]}{(n-1)(n-2)} & n>2 \\ \frac{1}{2\phi^2}\left[\log\frac{(\phi-x_{(1)})(\phi+x_{(2)})}{2\phi(x_{(2)}-x_{(1)})}\right] & n=2 \\ \frac{1}{2\phi^2}\left[x_{(1)}\log\frac{\phi-x_{(1)}}{\phi+x_{(1)}}+\phi\log\frac{(2\phi)^2}{(\phi-x_{(1)})(\phi+x_{(1)})}\right] & n=1, \end{cases}$$

which corresponds to (2.19).

A.2 Sign-Shifted-Exponential Distribution

Recall the definition of the sign-shifted exponential distribution in (2.4.4). Suppose we have n i.i.d. observations $x = (x_1, \ldots, x_n)$ that have each arisen from the $SSExp$ distribution. Then x will have a distribution related to (2.20), of the form

$$q(x|\theta) = \begin{cases} \prod_{i=1}^{n}\left[\omega e^{-\omega(x_i-a)}I_{(a,\infty)}(x_i)\right] & s=1 \\ \prod_{i=1}^{n}\left[\omega e^{\omega(x_i-a)}I_{(-\infty,a)}(x_i)\right] & s=-1, \end{cases} \tag{A.8}$$

where $\theta = (s, a, \omega)$ and s takes values from the set $S = \{-1, 1\}$, $a \in \mathbb{R}$ and $\omega > 0$.

Then, to find the marginal distribution, one requires to integrate out the parameters:

$$p(x|\phi) = \int q(x|\theta)\pi(\theta|\phi)\mathrm{d}\theta. \tag{A.9}$$

Consequently, a prior has to be defined. Suppose the prior on the parameters

(s, a, ω) has the following form:

$$\pi(\theta|\phi) = \pi(s, a, \omega|\phi) = \pi(s|\phi)\pi(a|\omega, \phi)\pi(\omega|\phi). \tag{A.10}$$

Let ω have a $Ga(\kappa, \beta)$ prior with κ and β known and both greater than 0, so that

$$\pi(\omega|\phi) = \frac{\beta^{\kappa}\omega^{\kappa-1}e^{-\beta\omega}}{\Gamma(\kappa)}. \tag{A.11}$$

Let a have a $Laplace(0, \gamma\omega)$ prior, where 0 is the mean parameter and $\gamma\omega$ is the scale parameter, and $\gamma > 0$:

$$\pi(a|\omega, \phi) = \frac{\gamma\omega}{2}e^{-\gamma\omega|a|}. \tag{A.12}$$

Hence,

$$\pi(a, \omega|\phi) = \pi(a|\omega, \phi)\pi(\omega|\phi) = \frac{\gamma\beta^{\kappa}\omega^{\kappa}e^{-\omega(\beta+\gamma|a|)}}{2\Gamma(\kappa)}. \tag{A.13}$$

Finally, we assume that s is *a priori* independent of a and ω,

$$\pi(s|a, \omega, \phi) = \begin{cases} \rho & s = 1 \\ (1-\rho) & s = -1, \end{cases} \tag{A.14}$$

where ρ is a hyperparameter which defines the probability of the sign parameter, and therefore $0 \leq \rho \leq 1$.

Now, since the prior has been defined the marginal distribution of the data can be evaluated. This will be carried out in two stages. Firstly, the marginal distribution of the data conditional on the sign parameter will be calculated, and

then these marginal distributions will be summed over the discrete distribution of the sign parameter to produce the marginal distribution of the data as required:

$$\begin{aligned} p(x|\phi) &= \sum_{S\in\{-1,1\}} p(x|s,\phi)\pi(s,\phi) \\ &= \rho p(x|s=1,\phi) + (1-\rho)p(-x|s=1,\phi). \end{aligned} \tag{A.15}$$

The marginal distribution of x conditional on the sign parameter being equal to 1 can be calculated in closed form as follows:

$$p(x|s=1,\phi) = \int\int p(x|s=1,a,\omega,\phi)\pi(a,\omega,\phi)\mathrm{d}a\mathrm{d}\omega, \tag{A.16}$$

and fortunately $p(x|s=-1,\phi)$ can be found from $p(x|s=1,\phi)$ because

$$\begin{aligned} p(x|s=-1,\phi) &= \int\int p(x|s=-1,a,\omega)\pi(a,\omega,\phi)\mathrm{d}a\mathrm{d}\omega & \text{(A.17)} \\ &= \int\int p(-x|s=1,-a,\omega)\pi(a,\omega,\phi)\mathrm{d}a\mathrm{d}\omega & \text{(A.18)} \\ &= \int\int p(-x|s=1,a,\omega)\pi(a,\omega,\phi)\mathrm{d}a\mathrm{d}\omega & \text{(A.19)} \\ &= p(-x|s=1,\phi). & \text{(A.20)} \end{aligned}$$

Note that to go from (A.17) to (A.18) one requires

$$\begin{aligned} p(x|s=-1,a,\omega) &= \omega e^{\omega(x-a)} I_{(-\infty,a)}(x) \\ &= \omega e^{-\omega(-x+a)} I_{(-a,\infty)}(-x) \\ &= p(-x|s=1,-a,\omega). \end{aligned}$$

The marginal distribution of the data conditional on $s=1$ requires the use of

$q(x|s=1,a,\omega)$ from (A.8) and $\pi(a,\omega|\phi)$ from (A.13):

$$
\begin{aligned}
p(x|s=1,\phi) &= \int\int q(x|s=1,a,\omega)\pi(a,\omega,\phi)\mathrm{d}a\mathrm{d}\omega \qquad -\infty<a<\infty,\ \ \omega>0,\ \ x>a \\
&= \int_0^\infty \int_{-\infty}^\infty \prod_{i=1}^n \left[\omega e^{-\omega(x_i-a)} I_{(a,\infty)}(x_i)\right] \frac{\gamma\beta^\kappa\omega^\kappa e^{-\omega(\beta+\gamma|a|)}}{2\Gamma(\kappa)}\mathrm{d}a\mathrm{d}\omega \qquad \text{(A.21)} \\
&= \int_0^\infty \int_{-\infty}^\infty \omega^n e^{-\omega\left(\sum_{i=1}^n x_i - na\right)} \frac{\gamma\beta^\kappa\omega^\kappa e^{-\omega(\beta+\gamma|a|)}}{2\Gamma(\kappa)} \prod_{i=1}^n I_{(-\infty,x_i)}(a)\mathrm{d}a\mathrm{d}\omega \qquad \text{(A.22)} \\
&= \frac{\gamma\beta^\kappa}{2\Gamma(\kappa)} \int_0^\infty \omega^{\kappa+n} \exp\left\{-\omega\left(\beta+\sum_{i=1}^n x_i\right)\right\} \cdot \\
&\qquad\qquad \int_{-\infty}^\infty \exp\left\{\omega(na-\gamma|a|)\right\} I_{(-\infty,x_{(1)})}(a)\mathrm{d}a\mathrm{d}\omega. \qquad \text{(A.23)}
\end{aligned}
$$

Note that in going from (A.21) to (A.22) one has to realise that

$$\prod_{i=1}^n I_{(a,\infty)}(x_i) = \prod_{i=1}^n I_{(-\infty,x_i)}(a).$$

Now, just consider the following integral from (A.23):

$$\int_{-\infty}^{\infty} \exp\left\{\omega(na-\gamma|a|)\right\} I_{(-\infty,x_{(1)})}(a)\mathrm{d}a \tag{A.24}$$

$$= \begin{cases} \int_0^{x_{(1)}} \exp\left\{\omega(n-\gamma)a\right\}\mathrm{d}a + \int_{-\infty}^{0} \exp\left\{\omega(n+\gamma)a\right\}\mathrm{d}a & x_{(1)} > 0 \\ \int_{-\infty}^{x_{(1)}} \exp\left\{\omega(n+\gamma)a\right\}\mathrm{d}a & x_{(1)} \leq 0 \end{cases}$$

$$= \int_{-\infty}^{\min(0,x_{(1)})} \exp\left\{\omega(n+\gamma)a\right\}\mathrm{d}a + \int_{0}^{\max(0,x_{(1)})} \exp\left\{\omega(n-\gamma)a\right\}\mathrm{d}a$$

$$= \begin{cases} \left[\frac{\exp\left\{\omega(n+\gamma)a\right\}}{\omega(n+\gamma)}\right]_{-\infty}^{\min(0,x_{(1)})} + \left[\frac{\exp\left\{\omega(n+\gamma)a\right\}}{\omega(n-\gamma)}\right]_{-\infty}^{\max(0,x_{(1)})} & n \neq \gamma \\ \left[\frac{\exp\left\{\omega(n+\gamma)a\right\}}{\omega(n+\gamma)}\right]_{-\infty}^{\min(0,x_{(1)})} + \max(0,x_{(1)}) & n = \gamma \end{cases}$$

$$= \begin{cases} \left[\frac{\exp\left\{\omega(n+\gamma)\min(0,x_{(1)})\right\}}{\omega(n+\gamma)}\right] + \left[\frac{\exp\left\{\omega(n+\gamma)\max(0,x_{(1)})\right\}-1}{\omega(n-\gamma)}\right] & n \neq \gamma \\ \left[\frac{\exp\left\{\omega(n+\gamma)\min(0,x_{(1)})\right\}}{\omega(n+\gamma)}\right] + \max(0,x_{(1)}) & n = \gamma. \end{cases} \tag{A.25}$$

Hence, substituting (A.25) into (A.23) produces two cases.

Let $C = \frac{\gamma\beta^\kappa}{2\Gamma(\kappa)}$. Then, for the case $n \neq \gamma$,

$$
\begin{aligned}
&p(x|s=1,\phi) \\
&= C\int_0^\infty \left(\omega^{\kappa+n}\exp\left\{-\omega\left(\beta+\sum_{i=1}^n x_i\right)\right\}\cdot\right. \\
&\qquad \left.\left[\frac{\exp\left\{\omega(n+\gamma)\min(0,x_{(1)})\right\}}{\omega(n+\gamma)}+\frac{\exp\left\{\omega(n+\gamma)\max(0,x_{(1)})\right\}-1}{\omega(n-\gamma)}\right]\right)\mathrm{d}\omega \\
&= C\left[\int_0^\infty \frac{\omega^{\kappa+n}\exp\left\{-\omega(\beta+\sum_{i=1}^n x_i)\right\}\exp\left\{\omega(n+\gamma)\min(0,x_{(1)})\right\}}{\omega(n+\gamma)}\mathrm{d}\omega\right. \\
&\qquad \left.+\int_0^\infty \frac{\omega^{\kappa+n}\exp\left\{-\omega(\beta+\sum_{i=1}^n x_i)\right\}\left(\exp\left\{\omega(n-\gamma)\max(0,x_{(1)})\right\}-1\right)}{\omega(n-\gamma)}\mathrm{d}\omega\right] \\
&= C\left[\frac{1}{n+\gamma}\int_0^\infty \omega^{\kappa+n-1}\exp\left\{-\omega(\beta+\sum_{i=1}^n x_i-(n+\gamma)\min(0,x_{(1)}))\right\}\mathrm{d}\omega\right. \\
&\qquad +\frac{1}{n-\gamma}\int_0^\infty \omega^{\kappa+n-1}\exp\left\{-\omega(\beta+\sum_{i=1}^n x_i-(n-\gamma)\max(0,x_{(1)}))\right\}\mathrm{d}\omega \\
&\qquad \left.-\frac{1}{n-\gamma}\int_0^\infty \omega^{\kappa+n-1}\exp\left\{-\omega(\beta+\sum_{i=1}^n x_i)\right\}\mathrm{d}\omega\right] \\
&= C\left[\frac{\Gamma(\kappa+n)}{(n+\gamma)\left[\beta+\sum_{i=1}^n x_i-(n+\gamma)\min(0,x_{(1)})\right]^{\kappa+n}}\right. \\
&\qquad +\frac{\Gamma(\kappa+n)}{(n-\gamma)\left[\beta+\sum_{i=1}^n x_i-(n-\gamma)\max(0,x_{(1)})\right]^{\kappa+n}} \\
&\qquad \left.-\frac{\Gamma(\kappa+n)}{(n-\gamma)\left[\beta+\sum_{i=1}^n x_i\right]^{\kappa+n}}\right]. \qquad \text{(A.26)}
\end{aligned}
$$

Note that $\int_0^\infty x^{t-1}e^{-ux}\mathrm{d}x = \frac{\Gamma(t)}{u^t}$ iff $t, u > 0$, otherwise the integral diverges. Hence, for the integral not to diverge we must have $(\kappa + n) > 0$, $(\beta + \sum_{i=1}^{n} x_i - (n + \gamma)\min(0, x_{(1)}) > 0)$, and $(\beta + \sum_{i=1}^{n} x - (n - \gamma)\max(0, x_{(1)}) > 0)$. All of these inequalities can easily be shown to hold.

Next, for the case when $n = \gamma$, we have

$$\begin{aligned} p(x|s=1,\phi) &= C\left[\frac{1}{n+\gamma}\int_0^\infty \omega^{\kappa+n-1}\exp\left\{-\omega(\beta+\sum_{i=1}^{n}x_i-(n+\gamma)\min(0,x_{(1)}))\right\}\mathrm{d}\omega\right. \\ &\quad \left.+max(0,x_{(1)})\int_0^\infty \omega^{\kappa+n}\exp\left\{-\omega(\beta+\sum_{i=1}^{n}x_i)\right\}\mathrm{d}\omega\right] \qquad \text{(A.27)} \\ &= C\left[\frac{1}{n+\gamma}\frac{\Gamma(\kappa+n)}{\left[\beta+\sum_{i=1}^{n}x_i-(n+\gamma)\min(0,x_{(1)})\right]^{\kappa+n}}\right. \\ &\quad \left.+max(0,x_{(1)})\frac{\Gamma(\kappa+n+1)}{\left[\beta+\sum_{i=1}^{n}x_i\right]^{\kappa+n+1}}\right], \qquad \text{(A.28)} \end{aligned}$$

where going from (A.27) to (A.28) follows directly from the results evaluated in the previous case where $n \neq \gamma$.

Thus, bringing (A.26) and (A.28) together results in

$$p(x|s=1,\phi)=\begin{cases}\frac{\gamma\beta^{\kappa}\Gamma(\kappa+n)}{2\Gamma(\kappa)}\Bigg(\frac{1}{(n+\gamma)(\beta+\sum_{i=1}^{n}x_i-(n+\gamma)\min(0,x_{(1)}))^{\kappa+n}} \\ \qquad +\frac{1}{(n-\gamma)(\beta+\sum_{i=1}^{n}x_i-(n+\gamma)\max(0,x_{(1)}))^{\kappa+n}} \\ \qquad -\frac{1}{(n-\gamma)(\beta+\sum_{i=1}^{n}x_i)^{\kappa+n}}\Bigg) & n\neq\gamma \\ \frac{\gamma\beta^{\kappa}}{2\Gamma(\kappa)}\Bigg(\frac{\Gamma(\kappa+n)}{(n+\gamma)(\beta+\sum_{i=1}^{n}x_i-(n+\gamma)\min(0,x_{(1)}))^{\kappa+n}}+\frac{\max(0,x_{(1)})\Gamma(\kappa+n+1)}{(\beta+\sum_{i=1}^{n}x_i)^{\kappa+n+1}}\Bigg) & n=\gamma.\end{cases} \tag{A.29}$$

Therefore, using (A.29), $p(x|s=-1,\phi)$ is obtained from (A.20):

$$p(x|s=-1,\phi)=\begin{cases}\frac{\gamma\beta^{\kappa}\Gamma(\kappa+n)}{2\Gamma(\kappa)}\Bigg(\frac{1}{(n+\gamma)(\beta-\sum_{i=1}^{n}x_i-(n+\gamma)\min(0,-x_{(n)}))^{\kappa+n}} \\ \qquad +\frac{1}{(n-\gamma)(\beta-\sum_{i=1}^{n}x_i-(n-\gamma)\max(0,-x_{(n)}))^{\kappa+n}} \\ \qquad -\frac{1}{(n-\gamma)(\beta-\sum_{i=1}^{n}x_i)^{\kappa+n}}\Bigg) & n\neq\gamma \\ \frac{\gamma\beta^{\kappa}}{2\Gamma(\kappa)}\Bigg(\frac{\Gamma(\kappa+n)}{(n+\gamma)(\beta-\sum_{i=1}^{n}x_i-(n+\gamma)\min(0,-x_{(n)}))^{\kappa+n}}+\frac{\max(0,-x_{(n)})\Gamma(\kappa+n+1)}{(\beta-\sum_{i=1}^{n}x_i)^{\kappa+n+1}}\Bigg) & n=\gamma.\end{cases} \tag{A.30}$$

Finally, the marginal distribution of the data can be calculated by inserting (A.29) and (A.30) into (A.15).

Appendix B

Calculation of effective sample size

This appendix contains the details of how the effective sample size, N_{eff}, used throughout the thesis is calculated.

From the allocation sampler comes a sample from the posterior distribution of k, but this is not an independent sample. The states of the Markov chain produced by the sampler are correlated because the next state of the chain is calculated using the current state of the chain. The effective sample size N_{eff} is the number of independent samples required to produce an estimate with the same precision as that given by N dependent samples. Then obviously $N_{eff} < N$. A way of estimating N_{eff} is

$$N_{eff} = \frac{N}{1 + 2\sum_{j=1}^{\infty} \rho_j}. \tag{B.1}$$

The problem now is how to estimate the quantity $\sum_{j=1}^{\infty} \rho_j$, which is the sum of the autocorrelations. One can think of the Markov chain as a stationary time series $X = X_1, \ldots, X_N$ with an autocorrelation function (acf) $\rho(j)$. An estimate of the sum of the acfs can be found by modelling the time series using an autoregressive model of order p that has coefficients $a(j)$:

$$X_t = \sum_{j=1}^{p} a(j) X_{t-j} + \varepsilon_t, \tag{B.2}$$

where ε_t is a Gaussian white noise process with zero mean and variance σ^2. The order of the AR model B.2 can be estimated by a number of different techniques, but the Akaike Information Criterion (AIC) method has been chosen here. Then, using results from the time series literature, see pages 18-56 of Granger and Newbold (1986) for details, one can state that

$$\sum_{i=-\infty}^{\infty} \rho(i) \;=\; \frac{1 - \sum_{j=1}^{p} a(j)\rho(j)}{\left[1 - \sum_{j=1}^{p} a(j)\right]^2}. \tag{B.3}$$

Thus, replacing B.3 as the denominator of B.1 one can calculate the effective sample size produced by the allocation sampler with respect to the number of components. For more details about effective sample sizes and their calculation see Geyer (1992).

Appendix C

Allocation sampler Fortran code

This appendix contains the full Fortran code required to produce the output files from which the posterior densities and estimates can be calculated. Where possible subroutines from the programs of Nobile (1994) were adapted. Only the program for the case where the components have a univariate normal density is presented. However, the majority of the subroutines given are also utilised when a different distribution is used for the components which is an advantage of this approach.

```
PROGRAM AllocationSampler

input files : file.data, file.par, file.sim, file.init (optional)
output files: file.log, file.k, file.g, file.relg (optional)

implicit none
integer KMAX,NMAX,HYPMAX,XSMAX,SAMPMAX
parameter (NMAX=2000,KMAX=50,HYPMAX=4,XSMAX=5,SAMPMAX=10000)
integer n,nsamp,burnin,thin,indinit,indlab,indsym,indrand,k,idg
integer km,dhyp,dxsm
integer g(NMAX),nj(KMAX),seed(4)
integer pos(KMAX),invpos(KMAX),firstav
integer i,oulog,ouk,oug,ougr,ouhyp,nfilnam,nsout
integer indgibbs,indmetr1,indmetr2,indmetr3,indejtabs,indmetlab
integer mtrprop(6),mtraccp(6),checksumm
double precision fk(KMAX),lfk(KMAX)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX)
double precision phi(HYPMAX,KMAX),xsumm(XSMAX,KMAX),const(KMAX)
double precision fggivkl,fxgivkgl,fglog
character fam
character*24 filename,filedat,filepar,filesim,fileinit
character*24 filelog,filek,fileg,filehyp

call filnams(filename,filedat,filepar,filesim,fileinit,filelog,
A             filek,fileg,filehyp,nfilnam)

call readdata(NMAX,n,x,filedat)

call readsim(SAMPMAX,nsamp,seed,burnin,thin,indinit,indlab,
A            indgibbs,indmetr1,indmetr2,indmetr3,indejtabs,
B            indmetlab,checksumm,filesim)

call readpar(KMAX,HYPMAX,km,dhyp,dxsm,indrand,indsym,fk,lfk,
A            alpha,alpha0,fam,phi,filepar)

call setseed(seed)

call getinit(KMAX,NMAX,indrand,indinit,k,km,n,g,indsym,fileinit,
A            nfilnam)

call setpos(KMAX,k,km,pos,invpos,firstav)

oulog = 1
ouk = 4
oug = 5
ougr = 8
ouhyp = 9
open(oulog,file=filelog,access='sequential',status='new')
open(ouk,file=filek,access='sequential',status='new')
open(oug,file=fileg,access='sequential',status='new')
if(indrand.eq.1) then
   open(ouhyp,file=filehyp,access='sequential',status='new')
endif

call writelogA(KMAX,NMAX,HYPMAX,k,km,n,g,dhyp,nsamp,seed,burnin,
A              thin,indinit,indlab,indrand,indsym,indgibbs,
B              indmetr1,indmetr2,indmetr3,indejtabs,indmetlab,
C              checksumm,fk,alpha,fam,phi,oulog)

call getconst(KMAX,HYPMAX,fam,km,phi,const)

call fkgx(KMAX,NMAX,HYPMAX,XSMAX,km,k,fam,lfk,n,g,pos,invpos,x,
A         alpha,alpha0,phi,nj,idg,xsumm,const,fggivkl,
B         fxgivkgl,fglog)

nsout=0
do i=1,6
   mtrprop(i)=0
   mtraccp(i)=0
enddo

if(indsym.eq.1.and.indlab.eq.1.and.(nsamp/thin).gt.SAMPMAX) then
   write(6,*) "Increase SAMPMAX in main program to", nsamp/thin
   stop
endif

do i=-burnin,nsamp

   if(indrand.eq.1) then
      call metrohyp(KMAX,NMAX,HYPMAX,XSMAX,i,burnin,km,n,x,k,
      A             fam,invpos,nj,xsumm,const,phi,fxgivkgl,fglog)
   endif

   call allocsamp(KMAX,NMAX,HYPMAX,XSMAX,km,k,fam,lfk,n,dxsm,g,
   A              pos,invpos,firstav,x,alpha,alpha0,phi,nj,idg,
   B              xsumm,const,fggivkl,fxgivkgl,fglog,indsym,
   C              indgibbs,indmetr1,indmetr2,indmetr3,
   D              indejtabs,indmetlab,mtrprop,mtraccp)

   if(checksumm.eq.1) then
      call summrychk(KMAX,NMAX,XSMAX,km,n,dxsm,k,fam,g,pos,
      A              invpos,x,nj,idg,xsumm)
   endif

   if(i.gt.0.and.mod(i,thin).eq.0) then
      if(indrand.eq.1) then
         call writehyp(KMAX,HYPMAX,fam,phi,ouhyp)
      endif
      call writekgout(KMAX,NMAX,k,n,g,pos,nj,idg,indsym,ouk,oug)
      nsout=nsout+1
   endif
enddo

call writelogB(KMAX,NMAX,k,n,g,pos,nj,idg,indsym,nsout,oulog,
A              mtrprop,mtraccp)

close(oulog,status='keep')
close(ouk,status='keep')
close(oug,status='keep')
if(indrand.eq.1) then
   close(ouhyp,status='keep')
endif

if(indsym.eq.1.and.indlab.eq.1) then
   call assign(NMAX,KMAX,SAMPMAX,km,n,nsout,ouk,oug,ougr,
   A           filek,fileg,nfilnam)
endif

stop
end


SUBROUTINE filnams(file1,filedat,filepar,filesim,fileinit,filelog,
A                  filek,fileg,filehyp,nfilnam)
implicit none
character*24 file1,filedat,filepar,filesim,fileinit,filelog
character*24 filek,fileg,filehyp
integer nfilnam

character*61 numb

logical filepres

integer i

data numb/"123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrst
A          uvwxyz"/

call getarg(1,file1)

do nfilnam = 24,1,-1
   if(file1(nfilnam:nfilnam).ne." ") go to 10
enddo
10   continue

filedat = file1(1:nfilnam)//".data"
filepar = file1(1:nfilnam)//".par"
filesim = file1(1:nfilnam)//".sim"
fileinit = file1(1:nfilnam)//".init"

inquire(file=filedat,exist=filepres)
if(.not.filepres) then
   write(6,*) "Error: ", filedat(1:nfilnam+5), " does not exist"
   stop
endif

inquire(file=filepar,exist=filepres)
if(.not.filepres) then
   write(6,*) "Error: ", filepar(1:nfilnam+4), " does not exist"
   stop
endif

inquire(file=filesim,exist=filepres)
if(.not.filepres) then
   write(6,*) "Error: ", filesim(1:nfilnam+4), " does not exist"
   stop
```

```
endif

do i=1,61
   filelog = file1(1:nfilnam)//"."//numb(i:i)//".log"
   inquire(file=filelog,exist=filepres)
   if(.not.filepres) then
      filek = file1(1:nfilnam)//"."//numb(i:i)//".k"
      fileg = file1(1:nfilnam)//"."//numb(i:i)//".g"
      filehyp = file1(1:nfilnam)//"."//numb(i:i)//".hyp"
      go to 20
    endif
enddo
write(6,*) "Error: too many runs made for ", file1(1:nfilnam)
write(6,*) "     Remove some of the .log, .k, .g, .hyp files"
stop
20   continue

return
end



SUBROUTINE readdata(NMAX,n,x,filedat)
implicit none
integer NMAX,n
double precision x(NMAX)
character*24 filedat

integer i

open(1,file=filedat,access='sequential',status='old')

read(1,*) n
read(1,*) (x(i),i=1,n)
close(1,status='keep')
return
end



SUBROUTINE findim(fam,dhyp,dxsm)
implicit none
character fam
integer dhyp,dxsm
if(fam.eq."N") then
   dhyp = 4
   dxsm = 2
   return
endif
write(6,*) "Wrong family in Sub. findim"
stop

end



SUBROUTINE readsim(SAMPMAX,nsamp,seed,burnin,thin,indinit,
A                  indlab,indgibbs,indmetr1,indmetr2,indmetr3,
B                  indejtabs,indmetlab,checksumm,filesim)
implicit none
integer SAMPMAX
integer nsamp,burnin,thin,indinit,indlab,seed(4)
integer indgibbs,indmetr1,indmetr2,indmetr3,indejtabs,indmetlab
integer checksumm
character*24 filesim

integer h,nsout

open(1,file=filesim,access='sequential',status='old')
read(1,*) nsamp
read(1,*) (seed(h),h=1,4)
read(1,*) burnin
read(1,*) thin
read(1,*) indinit
read(1,*) indlab
read(1,*) indgibbs
read(1,*) indmetr1
read(1,*) indmetr2
read(1,*) indmetr3
read(1,*) indejtabs
read(1,*) indmetlab
read(1,*) checksumm
close(1,status='keep')

nsout = nsamp/thin
if(indlab.eq.1.and.SAMPMAX.lt.nsout) then
   write(6,*) "SAMPMAX should be at least ", nsout
   stop
endif
if(indgibbs.ne.1.and.indmetr1.ne.1.and.indmetr2.ne.1.and.
A  indmetr3.ne.1.and.indejtabs.ne.1) then
   write(6,*) "At least one of "
   write(6,*) "   indgibbs,indmetr1,indmetr2,indmetr3,indejtabs"
   write(6,*) "   should be equal to 1"
   stop
endif

return
end



SUBROUTINE readpar(KMAX,HYPMAX,km,dhyp,dxsm,indrand,indsym,fk,
A                  lfk,alpha,alpha0,fam,phi,filepar)
implicit none
integer KMAX,HYPMAX
integer km,dhyp,dxsm,indrand,indsym
double precision fk(KMAX),lfk(KMAX),alpha(KMAX),alpha0(KMAX)
double precision phi(HYPMAX,KMAX)
character fam
character*24 filepar

integer i,j
double precision sumfk

open(1,file=filepar,access='sequential',status='old')

read(1,*) indrand
read(1,*) indsym
read(1,*) km

C Reads prior on k,then it normalizes to make them sum to 1

read(1,*) (fk(j),j=1,km)
sumfk = 0.0d-00
do j=1,km
   sumfk = sumfk + fk(j)
enddo
do j=1,km
   lfk(j) = dlog(fk(j) / sumfk)
enddo

read(1,1) fam
1    format(a1)
call findim(fam,dhyp,dxsm)

if (indsym.eq.0) then
   read(1,*) (alpha(j),j=1,km)
   alpha0(1)=alpha(1)
   do j=2,km
     alpha0(j)=alpha0(j-1)+alpha(j)
   enddo
   do j=1,km
     do i=1,dhyp
       read(1,*) phi(i,j)
     enddo
   enddo
else
   read(1,*) alpha(1)
   do i=1,dhyp
     read(1,*) phi(i,1)
   enddo
   do j=1,km
     alpha(j) = alpha(1)
     do i=1,dhyp
       phi(i,j) = phi(i,1)
     enddo
   enddo
   alpha0(1)=alpha(1)
   do j=2,km
     alpha0(j)=alpha0(j-1)+alpha(j)
   enddo
endif

close(1,status='keep')

return
```

```
end



SUBROUTINE writelogA(KMAX,NMAX,HYPMAX,k,km,n,g,dhyp,nsamp,seed,
A                    burnin,thin,indinit,indlab,indrand,indsym,
B                    indgibbs,indmetr1,indmetr2,indmetr3,
C                    indejtabs,indmetlab,checksumm,fk,alpha,fam,
D                    phi,oulog)

C This routine produces a files .log with
C exactly the same format as the .sim and .par files

implicit none
integer KMAX,NMAX,HYPMAX
integer k,km,n,dhyp,nsamp,burnin,thin,indinit,indlab
integer indgibbs,indmetr1,indmetr2,indmetr3,indejtabs
integer indmetlab,indrand,indsym,seed(4),oulog,checksumm
integer g(NMAX)
double precision fk(KMAX),alpha(KMAX)
double precision phi(HYPMAX,KMAX)
character fam

integer i,j

write(oulog,*) nsamp,                    "     nsamp"
write(oulog,*) (seed(i),i=1,4),          "     initseed"
write(oulog,*) burnin,                   "     burnin"
write(oulog,*) thin,                     "     thin"
write(oulog,*) indinit,                  "     indinit"
write(oulog,*) indlab,                   "     indlab"
write(oulog,*) indrand,                  "     indrand"
write(oulog,*) indsym,                   "     indsym"
write(oulog,*) indgibbs,                 "     indgibbs"
write(oulog,*) indmetr1,                 "     indmetr1"
write(oulog,*) indmetr2,                 "     indmetr2"
write(oulog,*) indmetr3,                 "     indmetr3"
write(oulog,*) indejtabs,                "     indejtabs"
write(oulog,*) indmetlab,                "     indmetlab"
write(oulog,*) checksumm,                "     checksumm"
write(oulog,*) km,                       "     km"
write(oulog,*) (fk(j),j=1,km),           "     fk"
write(oulog,*) fam,                      "     fam"
write(oulog,*) dhyp,                     "     dhyp"

if (indsym.eq.0) then
   write(oulog,*) (alpha(j),j=1,km),     "     alpha"
   do j=1,km
     do i=1,dhyp
         write(oulog,*) phi(i,j),        "     phi"
      enddo
   enddo
else
   write(oulog,*) alpha(1),              "     alpha"
   do i=1,dhyp
      write(oulog,*) phi(i,1),           "     phi"
   enddo
endif
write(oulog,*) k,                        "     k"
write(oulog,*) n,                        "     n"
write(oulog,*) (g(i),i=1,n),             "     g"

return
end



SUBROUTINE writelogB(KMAX,NMAX,k,n,g,pos,nj,idg,indsym,nsout,
A                    oulog,mtrprop,mtraccp)
implicit none
integer KMAX,NMAX
integer k,n,g(NMAX),pos(KMAX),nj(KMAX),idg,indsym
integer nsout,oulog,mtrprop(6),mtraccp(6)

integer i,seed(4),outg(NMAX)
character gchar(NMAX)

write(oulog,*) nsout,                    "     nsout"
write(oulog,*) mtrprop(1),mtraccp(1),    "     prpaccgibbs"
write(oulog,*) mtrprop(2),mtraccp(2),    "     prpaccmetr1"
write(oulog,*) mtrprop(3),mtraccp(3),    "     prpaccmetr2"
write(oulog,*) mtrprop(4),mtraccp(4),    "     prpaccmetr3"
write(oulog,*) mtrprop(5),mtraccp(5),    "     prpacceject"
write(oulog,*) mtrprop(6),mtraccp(6),    "     prpaccabsrb"

call getseed(seed)
write(oulog,*) (seed(i), i=1,4),         "     lastseed"
write(oulog,*) k,                        "     lastk"

if(indsym.eq.1) then
   call remgaps(KMAX,NMAX,k,n,g,pos,nj,idg,outg)
   write(oulog,*) (outg(i),i=1,n),       "     lastg"
else
   write(oulog,*) (g(i),i=1,n),          "     lastg"
endif

return
end



SUBROUTINE getinit(KMAX,NMAX,indrand,indinit,k,km,n,g,indsym,
A                  fileinit,nfilnam)
implicit none
integer KMAX,NMAX
integer indrand,indinit,k,km,n
integer g(NMAX),indsym,nfilnam
character*24 fileinit

integer i
double precision uni01kis

C If indrand = 1, forces initial k = km, g randomly generated
C otherwise, uses indinit to control starting values
C indinit = 1 sets k=1
C indinit = 2 random k, random g
C indinit = 3 reads k and g from .init file

if(indrand.eq.1) then
   k = km
   call randinit(KMAX,NMAX,k,km,n,g,indsym)
else
   if(indinit.eq.1) then
      k=1
      do i=1,n
         g(i) =1
      enddo
    endif
    if(indinit.eq.2) then
       k = int(km*UNI01KIS())+1
       call randinit(KMAX,NMAX,k,km,n,g,indsym)
    endif
    if(indinit.eq.3) then
       call readinit(KMAX,NMAX,k,km,n,g,indsym,fileinit,nfilnam)
    endif
endif

return
end



SUBROUTINE randinit(KMAX,NMAX,k,km,n,g,indsym)
implicit none
integer KMAX,NMAX
integer k,km,n,indsym
integer g(NMAX)

C Randomly generates a g vector

call gunif(NMAX,k,n,g)

call relabgnat(KMAX,NMAX,k,n,g)

return
end



SUBROUTINE gunif(NMAX,k,n,g)
implicit none
integer NMAX,k,n
integer g(NMAX)
double precision uni01kis

C Generates a n-vector of membership g with uniform distribution
```

```
C on the space of n-tuples consisting of elements of the set
C I={1,2,...,k}.

integer ind,i

do 10 i=1,n
   ind=int(k*UNI01KIS())+1
   g(i)=ind
10   continue

return
end



SUBROUTINE readinit(KMAX,NMAX,k,km,n,g,indsym,fileinit,nfilnam)
implicit none
integer KMAX,NMAX
integer k,km,n
integer g(NMAX),indsym,nfilnam
character*24 fileinit

integer i
logical filepres

inquire(file=fileinit,exist=filepres)
if(.not.filepres) then
   write(6,*) "Error: ",fileinit(1:nfilnam+5)," does not exist"
   stop
endif

open(1,file=fileinit,access='sequential',status='old')
read(1,*) k
read(1,*) (g(i),i=1,n)
close(1,status='keep')

if (k.gt.km) then
   write(6,*) "Error: k is larger than km"
   stop
endif

do i=1,n
   if (g(i).gt.k) then
      write(6,*) "Error: g vector has element > k"
      stop
    endif
enddo

if (indsym.eq.1) then
  call relabgnat(KMAX,NMAX,k,n,g)
endif

return
end



SUBROUTINE setpos(KMAX,k,km,pos,invpos,firstav)
implicit none
integer KMAX
integer k,km,firstav
integer pos(KMAX),invpos(KMAX)

integer j

do j=1,k
   pos(j) = j
   invpos(j) = j
enddo
firstav = k+1
if(k.lt.km) then
   do j=k+1,km
      pos(j)=0
      invpos(j)=0
   enddo
endif

return
end



SUBROUTINE getconst(KMAX,HYPMAX,fam,km,phi,const)
implicit none
integer KMAX,HYPMAX
integer km
double precision phi(HYPMAX,KMAX)
double precision const(KMAX)
character fam

if(fam.eq."N") then
   call getconstN(KMAX,HYPMAX,km,phi,const)
   return
endif
write(6,*) "Wrong family in Sub. getconst"
stop

end



SUBROUTINE allocsamp(KMAX,NMAX,HYPMAX,XSMAX,km,k,fam,lfk,n,dxsm,g,
A                    pos,invpos,firstav,x,alpha,alpha0,phi,nj,idg,
B                    xsumm,const,fggivkl,fxgivkgl,fglog,indsym,
C                    indgibbs,indmetr1,indmetr2,indmetr3,
D                    indejtabs,indmetlab,mtrprop,mtraccp)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer km,k,n,dxsm,firstav,idg,indsym
integer g(NMAX),nj(KMAX),pos(KMAX),invpos(KMAX)
integer indgibbs,indmetr1,indmetr2,indmetr3,indejtabs,indmetlab
integer mtrprop(6),mtraccp(6)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX),lfk(KMAX)
double precision fggivkl,fxgivkgl,fglog
double precision xsumm(XSMAX,KMAX),const(KMAX),phi(HYPMAX,KMAX)
character fam

integer j,done,indgm
double precision uni01kis,runif,ejectp(KMAX)

ejectp(1) = 1.0d-00
ejectp(km) = 0.0d-00
do j=2,km-1
   ejectp(j) = 0.5d-00
enddo

indgm = indgibbs + indmetr1 + indmetr2 + indmetr3
done = 0
10   continue
runif = UNI01KIS()
if(runif.lt.0.5d-00.and.indgm.gt.0) then
20   continue
   runif = UNI01KIS()
   if(runif.lt.0.25d-00) then
      if(indgibbs.eq.1) then
         call gibbs(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,pos,
         A          invpos,x,nj,idg,fggivkl,fxgivkgl,fglog,xsumm,
         B          const,alpha,alpha0,phi,mtrprop,mtraccp)

         done = 1
      endif
   else if(runif.lt.0.5d-00) then
      if(indmetr1.eq.1) then
         call metrstep1(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,
         A              pos,invpos,x,nj,idg,fggivkl,fxgivkgl,
         B              fglog,xsumm,const,alpha,alpha0,phi,
         C              mtrprop,mtraccp)
         done = 1
       endif
   else if(runif.lt.0.75d-00) then
      if(indmetr2.eq.1) then
         call metrstep2(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,
         A              pos,invpos,x,nj,idg,fggivkl,fxgivkgl,
         B              fglog,xsumm,const,alpha,alpha0,phi,
         C              mtrprop,mtraccp)
         done = 1
      endif
   else
      if(indmetr3.eq.1) then
         call metrstep3(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,
         A              pos,invpos,x,nj,idg,fggivkl,fxgivkgl,
         B              fglog,xsumm,const,alpha,alpha0,phi,
         C              mtrprop,mtraccp)
         done = 1
      endif
   endif
```

```
   if(done.eq.0) go to 20
else
   if(indejtabs.eq.1) then
      call ejtabs(KMAX,NMAX,HYPMAX,XSMAX,km,n,k,fam,lfk,dxsm,g,
      A           pos,invpos,firstav,x,nj,idg,fggivkl,fxgivkgl,
      B           fglog,xsumm,const,alpha,alpha0,phi,indsym,
      C           ejectp,mtrprop,mtraccp)
      done = 1
   endif
endif
if(done.eq.0) go to 10

if(indmetlab.eq.1) then
   call metrolab(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,g,pos,
   A             invpos,nj,fggivkl,fxgivkgl,fglog,xsumm,
   B             const,alpha,phi,indsym)
endif

return
end



SUBROUTINE gibbs(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,pos,
A                invpos,x,nj,idg,fggivkl,fxgivkgl,fglog,xsumm,
B                const,alpha,alpha0,phi,mtrprop,mtraccp)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer n,k,dxsm
integer g(NMAX),idg,nj(KMAX),pos(KMAX),invpos(KMAX)
integer mtrprop(6),mtraccp(6)
double precision fglog,fggivkl,fxgivkgl
double precision x(NMAX),xsumm(XSMAX,KMAX),const(KMAX)
double precision alpha(KMAX)alpha0(KMAX)
double precision phi(HYPMAX,KMAX),lfk(KMAX)
character fam

C Selects a new value NEWGI, in place of the old OLDGI,
C              for the component G_{I}.

integer j,idgn,nj1,nj2,p,h
integer i,oldgi,newgi
double precision lfxggivk(KMAX),prob(KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX)
double precision fggivkln,fxgivkgn,fgnlog

if(k.gt.1) then
   mtrprop(1) = mtrprop(1) + 1
   mtraccp(1) = mtraccp(1) + 1
   do i=1,n

C Computing f{X,G|K} (g_[1:i-1], j, g_[i+1:n])

      oldgi=g(i)
      do p=1,k
         j=invpos(p)
         if(j.eq.oldgi) then
            lfxggivk(p)=fglog
         else
            call fkgx1(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,i,
            A          oldgi,j,pos,nj,x,g,xsumm,const,idg,idgn,
            B          nj1,nj2,xsumm1,xsumm2,alpha,phi,fggivkl,
            C          fggivkln,fxgivkgl,fxgivkgn,fgnlog)
            lfxggivk(p)=fgnlog
         endif
      enddo

C Computing the probabilities of modifying g(i) to j

      call comprob(KMAX,k,lfxggivk,prob)

C Selecting the component's new value

      call selecomp(KMAX,k,oldgi,newgi,pos,invpos,prob)

C Update g

      if(oldgi.ne.newgi) then
         call fkgx1(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,i,
         A          oldgi,newgi,pos,nj,x,g,xsumm,const,idg,idgn,
         B          nj1,nj2,xsumm1,xsumm2,alpha,phi,fggivkl,
         C          fggivkln,fxgivkgl,fxgivkgn,fgnlog)
         call updateg(KMAX,NMAX,XSMAX,dxsm,i,oldgi,newgi,pos,
         A            g,nj,idg,idgn,nj1,nj2,xsumm,xsumm1,
         B            xsumm2,fggivkl,fxgivkgl,fglog,fggivkln,
         C            fxgivkgn,fgnlog)
      endif

   enddo
endif

return
end



SUBROUTINE comprob(KMAX,k,lfxggivk,prob)
implicit none
integer KMAX,k
double precision lfxggivk(KMAX),prob(KMAX)

integer p
double precision lmax

call maxvec(KMAX,k,lfxggivk,lmax)
do p=1,k
   prob(p)=dexp(lfxggivk(p)-lmax)
enddo

C Cumulate and normalize

do p=2,k
   prob(p)=prob(p-1)+prob(p)
enddo
do p=1,k
   prob(p)=prob(p)/prob(k)
enddo

return
end



SUBROUTINE selecomp(KMAX,k,oldgi,newgi,pos,invpos,prob)
implicit none
integer KMAX,k,oldgi,newgi,pos(KMAX),invpos(KMAX)
double precision prob(KMAX)

integer p,pold,pnew
double precision prob1,runif
double precision uni01kis

pold=pos(oldgi)

if(pold.eq.1) then
   prob1=0.0d-00
else
  prob1=prob(pold-1)
endif

runif=UNI01KIS()
if(runif.gt.prob1.and.runif.le.prob(pold)) then
   pnew=pold
else
   if(pold.gt.1) then
      pnew=1
      do p=1,pold-1
         if(runif.le.prob(p)) go to 10
         pnew=pnew+1
      enddo
   endif
   if(pold.lt.k) then
      pnew=pold+1
      do p=pold+1,k
         if(runif.le.prob(p)) go to 10
         pnew=pnew+1
      enddo
   endif
10      continue
endif

newgi=invpos(pnew)

return
end
```

```
SUBROUTINE maxvec(KMAX,k,vec,maxv)
implicit none
integer KMAX,k
double precision vec(KMAX),maxv

integer i

maxv = vec(1)

do i=2,k
   if (vec(i).gt.maxv) maxv = vec(i)
enddo

return
end



SUBROUTINE metrstep1(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,
A                    g,pos,invpos,x,nj,idg,fggivkl,fxgivkgl,
B                    fglog,xsumm,const,alpha,alpha0,phi,mtrprop,
C                    mtraccp)
implicit none
integer NMAX,KMAX,HYPMAX,XSMAX
integer n,k,dxsm
integer g(NMAX),nj(KMAX),idg,pos(KMAX),invpos(KMAX)
integer mtrprop(6),mtraccp(6)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX)
double precision fggivkl,fxgivkgl,fglog,lfk(KMAX)
double precision xsumm(XSMAX,KMAX),const(KMAX),phi(HYPMAX,KMAX)
character fam

integer p1,p2,j1,j2,nj1,nj2,i,j,g1(NMAX)
double precision uni01kis,gamdev,prob,lrn
double precision xsumm1(XSMAX),xsumm2(XSMAX)
double precision fggivkl1,fxgivkgl1,fg1log

if(k.eq.1) return

p1 = int(k*UNI01KIS()) + 1
10   continue
p2 = int(k*UNI01KIS()) + 1
if(p2.eq.p1) go to 10

if(nj(p1).eq.0.and.nj(p2).eq.0) return

mtrprop(2) = mtrprop(2) + 1
j1 = invpos(p1)
j2 = invpos(p2)
call zeromet(XSMAX,dxsm,nj1,nj2,xsumm1,xsumm2)

prob = 1.0d-00 / (1.0d-00+gamdev(alpha(j2))/gamdev(alpha(j1)))

do i=1,n
   if(g(i).eq.j1.or.g(i).eq.j2) then
      if(UNI01KIS().lt.prob) then
         j=j1
      else
         j=j2
      endif
      g1(i) = j
      call summetr(NMAX,XSMAX,fam,i,j,j1,nj1,nj2,x,xsumm1,
      A            xsumm2)
   endif
enddo

call fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A             j1,j2,p1,p2,nj1,nj2,xsumm1,xsumm2,
B             fxgivkgl,fxgivkgl1)

lrn=dlog(UNI01KIS())
if((fxgivkgl1 - fxgivkgl).gt.lrn) then
   mtraccp(2) = mtraccp(2) + 1
   call fggivkmet(KMAX,k,alpha,alpha0,j1,j2,p1,p2,nj1,nj2,n,n,
   A              nj,fggivkl,fggivkl1)
   fg1log = lfk(k) + fggivkl1 + fxgivkgl1
   call updatmet(NMAX,KMAX,XSMAX,dxsm,n,j1,j2,p1,p2,g,g1,nj,
   A             nj1,nj2,idg,xsumm,xsumm1,xsumm2,fggivkl,
   B             fxgivkgl,fglog,fggivkl1,fxgivkgl1,fg1log)
endif

return
end



SUBROUTINE zeromet(XSMAX,dxsm,nj1,nj2,xsumm1,xsumm2)
implicit none
integer XSMAX,dxsm,nj1,nj2
double precision xsumm1(XSMAX),xsumm2(XSMAX)
integer m

nj1 = 0
nj2 = 0
do m=1,dxsm
   xsumm1(m) = 0.0d-00
   xsumm2(m) = 0.0d-00
enddo

return
end



SUBROUTINE summetr(NMAX,XSMAX,fam,i,j,j1,nj1,nj2,x,xsumm1,xsumm2)
implicit none
integer NMAX,XSMAX,i,j,j1,nj1,nj2
double precision x(NMAX),xsumm1(XSMAX),xsumm2(XSMAX)
character fam

if(fam.eq."N") then
   call summetrN(NMAX,XSMAX,i,j,j1,nj1,nj2,x,xsumm1,xsumm2)
   return
endif
write(6,*) "Wrong family in Sub. summetr"
stop

end



SUBROUTINE fggivkmet(KMAX,k,alpha,alpha0,j1,j2,p1,p2,nj1,nj2,n,n1,
A                    nj,fggivkl,fggivkl1)
implicit none
integer KMAX,k,j1,j2,p1,p2,nj1,nj2,n,n1,nj(KMAX)
double precision alpha(KMAX),alpha0(KMAX),fggivkl,fggivkl1

double precision alph0n,alphj1,alphj2
double precision gam1,gam2,gam3,gam4,gam5,gam6

alph0n = alpha0(k) + n
call dlgama(alph0n,gam1)
alph0n = alpha0(k) + n1
call dlgama(alph0n,gam2)
alphj1 = alpha(j1) + dfloat(nj(p1))
alphj2 = alpha(j2) + dfloat(nj(p2))
call dlgama(alphj1,gam3)
call dlgama(alphj2,gam4)
alphj1 = alpha(j1) + dfloat(nj1)
alphj2 = alpha(j2) + dfloat(nj2)
call dlgama(alphj1,gam5)
call dlgama(alphj2,gam6)
fggivkl1 = fggivkl + gam1 - gam2 - gam3 - gam4 + gam5 + gam6

return
end



SUBROUTINE updatmet(NMAX,KMAX,XSMAX,dxsm,n,j1,j2,p1,p2,g,g1,nj,
A                   nj1,nj2,idg,xsumm,xsumm1,xsumm2,fggivkl,
B                   fxgivkgl,fglog,fggivkl1,fxgivkgl1,fg1log)
implicit none
integer NMAX,KMAX,XSMAX
integer dxsm,n,j1,j2,p1,p2,nj1,nj2,idg
integer g(NMAX),g1(NMAX),nj(KMAX)
double precision xsumm(XSMAX,KMAX),xsumm1(XSMAX),xsumm2(XSMAX)
double precision fggivkl,fxgivkgl,fglog,fggivkl1,fxgivkgl1,fg1log

integer i,m

do i=1,n
   if(g(i).eq.j1.or.g(i).eq.j2) g(i) = g1(i)
enddo
```

```
if(nj(p1).gt.0.and.nj1.eq.0) idg=idg-1
if(nj(p1).eq.0.and.nj1.gt.0) idg=idg+1
if(nj(p2).gt.0.and.nj2.eq.0) idg=idg-1
if(nj(p2).eq.0.and.nj2.gt.0) idg=idg+1
nj(p1) = nj1
nj(p2) = nj2

do m=1,dxsm
   xsumm(m,p1) = xsumm1(m)
   xsumm(m,p2) = xsumm2(m)
enddo

fggivkl = fggivkl1
fxgivkgl = fxgivkgl1
fglog = fg1log

return
end



SUBROUTINE fkgx(KMAX,NMAX,HYPMAX,XSMAX,km,k,fam,lfk,n,g,pos,
A               invpos,x,alpha,alpha0,phi,nj,idg,xsumm,const,
B               fggivkl,fxgivkgl,fglog)
implicit none
integer NMAX,KMAX,HYPMAX,XSMAX
integer km,k,n
integer g(NMAX),nj(KMAX),pos(KMAX),invpos(KMAX),idg
double precision x(NMAX),alpha(KMAX),alpha0(KMAX)
double precision xsumm(XSMAX,KMAX),lfk(KMAX)
double precision phi(HYPMAX,KMAX),const(KMAX)
double precision fggivkl,fxgivkgl,fglog
character fam
integer i

C Computes log f_{K,G,X} = log [f_{K} f_{G|K} f_{X|K,G} ]

call selecxg(KMAX,NMAX,XSMAX,km,n,k,fam,g,pos,x,nj,xsumm,idg)
call fggivk(KMAX,n,k,invpos,nj,alpha,alpha0,fggivkl)
call fxgivkg(KMAX,HYPMAX,XSMAX,n,k,fam,invpos,nj,xsumm,const,
A            phi,fxgivkgl)
fglog=lfk(k)+fggivkl+fxgivkgl

return
end



SUBROUTINE selecxg(KMAX,NMAX,XSMAX,km,n,k,fam,g,pos,x,nj,
A                  xsumm,idg)
implicit none
integer KMAX,NMAX,XSMAX
integer km,k,n
integer g(NMAX),nj(KMAX),pos(KMAX),idg
double precision x(NMAX),xsumm(XSMAX,KMAX)
character fam

if(fam.eq."N") then
   call selecxgN(KMAX,NMAX,XSMAX,km,n,k,g,pos,x,nj,xsumm,idg)
   return
endif
write(6,*) "Wrong family in Sub. selecxg"
stop

end



SUBROUTINE fggivk(KMAX,n,k,invpos,nj,alpha,alpha0,fggivkl)
implicit none
integer KMAX,n,k
integer invpos(KMAX),nj(KMAX)
double precision alpha(KMAX),alpha0(KMAX),fggivkl

C Computes    log f_(G|K)

integer j,p
double precision sumalph,sumalpn,gam1,gam2,alphaj,alphjn

sumalph=alpha0(k)
sumalpn=sumalph+dfloat(n)
call dlgama(sumalph,gam1)
call dlgama(sumalpn,gam2)
fggivkl=gam1-gam2
do p=1,k
   j=invpos(p)
   alphaj=alpha(j)
   alphjn=alphaj+dfloat(nj(p))
   call dlgama(alphjn,gam1)
   call dlgama(alphaj,gam2)
   fggivkl=fggivkl+gam1-gam2
enddo

return
end



SUBROUTINE fxgivkg(KMAX,HYPMAX,XSMAX,n,k,fam,invpos,nj,xsumm,
A                  const,phi,fxgivkgl)
implicit none
integer KMAX,HYPMAX,XSMAX
integer n,k
integer invpos(KMAX),nj(KMAX)
double precision xsumm(XSMAX,KMAX),fxgivkgl
double precision phi(HYPMAX,KMAX),const(KMAX)
character fam

if(fam.eq."N") then
   call fxgivkgN(KMAX,HYPMAX,XSMAX,n,k,invpos,nj,xsumm,const,phi,
   A             fxgivkgl)
   return
endif
write(6,*) "Wrong family in Sub. fxgivkg"
stop

end



SUBROUTINE metrohyp(KMAX,NMAX,HYPMAX,XSMAX,i,burnin,km,n,x,k,
A                   fam,invpos,nj,xsumm,const,phi,fxgivkgl,fglog)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer i,burnin,km,n,k,invpos(KMAX),nj(KMAX)
double precision x(NMAX),xsumm(XSMAX,KMAX),const(KMAX)
double precision phi(HYPMAX,KMAX),fxgivkgl,fglog
character fam

if(fam.eq."N") then
   call metrohypN(KMAX,NMAX,HYPMAX,XSMAX,i,burnin,km,n,x,k,
   A              invpos,nj,xsumm,const,phi,fxgivkgl,fglog)
   return
endif
write(6,*) "Wrong family in Sub. metrohyp"
stop

end



SUBROUTINE fkgx1(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,igcomp,oldgi,
A                newgi,pos,nj,x,g,xsumm,const,idg,idgn,nj1,nj2,
B                xsumm1,xsumm2,alpha,phi,fggivkl,fggivkln,
C                fxgivkgl,fxgivkgn,fgnlog)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer n,k,igcomp,oldgi,newgi,pos(KMAX)
integer g(NMAX),nj(KMAX),idg,idgn,nj1,nj2
double precision x(NMAX),xsumm(XSMAX,KMAX),const(KMAX),lfk(KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX),alpha(KMAX)
double precision phi(HYPMAX,KMAX)
double precision fggivkl,fggivkln,fxgivkgl,fxgivkgn,fgnlog
character fam

integer pold,pnew,h

pold = pos(oldgi)
pnew = pos(newgi)

call selecxg1(KMAX,NMAX,HYPMAX,XSMAX,fam,n,x,g,igcomp,oldgi,newgi,
A             pold,pnew,nj,xsumm,idg,idgn,nj1,nj2,xsumm1,xsumm2)
call fggivk1(KMAX,alpha,oldgi,newgi,nj1,nj2,fggivkl,fggivkln)
call fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A             oldgi,newgi,pold,pnew,nj1,nj2,xsumm1,xsumm2,
B             fxgivkgl,fxgivkgn)
```

```
fgnlog=lfk(k)+fggivkln+fxgivkgn

return
end



SUBROUTINE selecxg1(KMAX,NMAX,HYPMAX,XSMAX,fam,n,x,g,igcomp,
A                   oldgi,newgi,pold,pnew,nj,xsumm,idg,idgn,
B                   nj1,nj2,xsumm1,xsumm2)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer n,igcomp,oldgi,newgi,pold,pnew
integer g(NMAX),nj(KMAX),idg,idgn,nj1,nj2
double precision x(NMAX),xsumm(XSMAX,KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX)
character fam

if(fam.eq."N") then
   call selecxg1N(KMAX,NMAX,HYPMAX,XSMAX,x,igcomp,oldgi,newgi,
   A              pold,pnew,nj,xsumm,idg,idgn,nj1,nj2,xsumm1,
   B              xsumm2)
   return
endif
write(6,*) "Wrong family in Sub. selecxg1"
stop

end



SUBROUTINE fggivk1(KMAX,alpha,oldgi,newgi,nj1,nj2,fggivkl,
A                  fggivkln)
implicit none
integer KMAX,oldgi,newgi,nj1,nj2
double precision alpha(KMAX),fggivkl,fggivkln

fggivkln=fggivkl+dlog(alpha(newgi)+dfloat(nj2-1))
1                     -dlog(alpha(oldgi)+dfloat(nj1))

return
end



SUBROUTINE fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A                   oldgi,newgi,pold,pnew,nj1,nj2,xsumm1,xsumm2,
B                   fxgivkgl,fxgivkgn)
implicit none
integer KMAX,HYPMAX,XSMAX
integer nj(KMAX),oldgi,newgi,nj1,nj2,pold,pnew
double precision xsumm(XSMAX,KMAX),const(KMAX),xsumm1(XSMAX)
double precision xsumm2(XSMAX),phi(HYPMAX,KMAX)
double precision fxgivkgl,fxgivkgn
character fam

if(fam.eq."N") then
   call fxgivkg1N(KMAX,HYPMAX,XSMAX,nj,xsumm,const,phi,oldgi,
   A              newgi,pold,pnew,nj1,nj2,xsumm1,xsumm2,fxgivkgl,
   B              fxgivkgl,fxgivkgn)
   return
endif
write(6,*) "Wrong family in Sub. fxgivkg1"
stop

end



SUBROUTINE updateg(KMAX,NMAX,XSMAX,dxsm,igcomp,oldgi,newgi,pos,
A                  g,nj,idg,idgn,nj1,nj2,xsumm,xsumm1,xsumm2,
B                  fggivkl,fxgivkgl,fglog,fggivkln,fxgivkgn,
C                  fgnlog)
implicit none
integer KMAX,NMAX,XSMAX
integer dxsm,igcomp,oldgi,newgi
integer g(NMAX),nj(KMAX),idg,idgn,nj1,nj2,pos(KMAX)
double precision xsumm(XSMAX,KMAX),xsumm1(XSMAX),xsumm2(XSMAX)
double precision fggivkl,fggivkln,fxgivkgl,fxgivkgn
double precision fglog,fgnlog

integer i,pold,pnew

pold=pos(oldgi)
pnew=pos(newgi)

g(igcomp)=newgi
idg=idgn
nj(pold)=nj1
nj(pnew)=nj2
do i=1,dxsm
   xsumm(i,pold)=xsumm1(i)
   xsumm(i,pnew)=xsumm2(i)
enddo

fggivkl=fggivkln
fxgivkgl=fxgivkgn
fglog=fgnlog

return
end



SUBROUTINE metrstep2(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,
A                    pos,invpos,x,nj,idg,fggivkl,fxgivkgl,fglog,
B                    xsumm,const,alpha,alpha0,phi,mtrprop,mtraccp)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer n,k,dxsm
integer g(NMAX),idg,nj(KMAX),pos(KMAX),invpos(KMAX)
integer mtrprop(6),mtraccp(6)
double precision fglog,fggivkl,fxgivkgl
double precision x(NMAX),xsumm(XSMAX,KMAX),const(KMAX),alpha(KMAX)
double precision alpha0(KMAX),phi(HYPMAX,KMAX),lfk(KMAX)
character fam

integer p,j,njmove,i,count,ind,njj,ij,pnew,jnew,idgn,nj1,nj2
integer index(NMAX), gmove(NMAX)
cdouble precision lfxggivk(KMAX),prob(KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX)
double precision fggivkln,fxgivkgn,fgnlog
double precision dn,gam1,gam2,gam3,gam4,lograt
double precision uni01kis

if(k.eq.1) return

C Select component
p = int(k*UNI01KIS()) + 1
if(nj(p).eq.0) return
j = invpos(p)

C njmove: number of observations in component j that change
njmove = int(nj(p) * UNI01KIS()) + 1

do count=1,nj(p)
   index(count) = 0
enddo
count = 0
10   continue
ind = int(nj(p) * UNI01KIS()) + 1
if(index(ind).eq.0) then
   index(ind) = 1
   count = count + 1
   if(count.eq.njmove) go to 20
endif
go to 10
20   continue

count = 0
do i=1,nj(p)
   if(index(i).eq.1) then
      count = count + 1
      index(count) = i
      if(count.eq.njmove) go to 30
   endif
enddo
30   continue

i = 0
njj = 0
do count=1,njmove
   ij = index(count)
40      continue
   i = i + 1
```

```
   if(g(i).eq.j) then
      njj = njj + 1
      if(ij.eq.njj) then
         gmove(count) = i
         go to 50
      endif
   endif
   go to 40
50      continue
enddo

60   continue
pnew = int(k*UNI01KIS()) + 1
if(pnew.eq.p) go to 60
jnew=invpos(pnew)
call fkgx2(KMAX,NMAX,HYPMAX,XSMAX,k,n,fam,lfk,njmove,gmove,j,
A          jnew,pos,nj,x,g,xsumm,const,idg,idgn,nj1,nj2,xsumm1,
B          xsumm2,alpha,phi,fggivkl,fggivkln,fxgivkgl,fxgivkgn,
C          fgnlog)
lograt = fgnlog - fglog
dn = dfloat(nj(p) + 1)
call dlgama(dn, gam1)
dn = dfloat(nj(p) - njmove + 1)
call dlgama(dn, gam2)
dn = dfloat(nj2 + 1)
call dlgama(dn, gam3)
dn = dfloat(nj2 - njmove)
call dlgama(dn, gam4)
lograt = lograt + gam1 - gam2 - gam3 + gam4
A        + dlog(dfloat(nj(p))) - dlog(dfloat(nj2 - njmove))
if(lograt.gt.dlog(UNI01KIS())) then
   mtraccp(3) = mtraccp(3) + 1
   call updateg2(KMAX,NMAX,XSMAX,dxsm,njmove,gmove,j,jnew,pos,
   A             g,nj,idg,idgn,nj1,nj2,xsumm,xsumm1,xsumm2,
   B             fggivkl,fxgivkgl,fglog,fggivkln,fxgivkgn,
   C             fgnlog)
   endif

return
end



SUBROUTINE fkgx2(KMAX,NMAX,HYPMAX,XSMAX,k,n,fam,lfk,njmove,
A                gmove,jold,jnew,pos,nj,x,g,xsumm,const,idg,
B                idgn,nj1,nj2,xsumm1,xsumm2,alpha,phi,fggivkl,
C                fggivkln,fxgivkgl,fxgivkgn,fgnlog)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer k,n,njmove,gmove(NMAX),jold,jnew,pos(KMAX)
integer g(NMAX),nj(KMAX),idg,idgn,nj1,nj2
double precision x(NMAX),xsumm(XSMAX,KMAX),const(KMAX),lfk(KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX),alpha(KMAX)
double precision phi(HYPMAX,KMAX)
double precision fggivkl,fggivkln,fxgivkgl,fxgivkgn,fgnlog
character fam

integer pold,pnew

pold = pos(jold)
pnew = pos(jnew)
call selecxg2(KMAX,NMAX,HYPMAX,XSMAX,fam,n,x,g,njmove,gmove,
A             jold,jnew,pold,pnew,nj,xsumm,idg,idgn,nj1,nj2,
B             xsumm1,xsumm2)
call fggivk2(KMAX,alpha,jold,jnew,njmove,nj1,nj2,fggivkl,
A            fggivkln)
call fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,jold,
A             jnew,pold,pnew,nj1,nj2,xsumm1,xsumm2,fxgivkgl,
B             fxgivkgn)

fgnlog=lfk(k)+fggivkln+fxgivkgn

return
end



SUBROUTINE selecxg2(KMAX,NMAX,HYPMAX,XSMAX,fam,n,x,g,njmove,
A                   gmove,jold,jnew,pold,pnew,nj,xsumm,idg,idgn,
B                   nj1,nj2,xsumm1,xsumm2)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer n,njmove,gmove(NMAX),jold,jnew,pold,pnew
integer g(NMAX),nj(KMAX),idg,idgn,nj1,nj2
double precision x(NMAX),xsumm(XSMAX,KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX)
character fam

if(fam.eq."N") then
   call selecxg2N(KMAX,NMAX,HYPMAX,XSMAX,x,njmove,gmove,jold,
   A              jnew,pold,pnew,nj,xsumm,idg,idgn,nj1,nj2,
   B              xsumm1,xsumm2)
   return
endif
write(6,*) "Wrong family in Sub. selecxg2"
stop

end



SUBROUTINE fggivk2(KMAX,alpha,jold,jnew,njmove,nj1,nj2,fggivkl,
A                  fggivkln)
implicit none
integer KMAX,jold,jnew,njmove,nj1,nj2
double precision alpha(KMAX),fggivkl,fggivkln

double precision alphan,gam1,gam2,gam3,gam4

alphan = alpha(jold) + nj1
call dlgama(alphan, gam1)
alphan = alpha(jold) + nj1 + njmove
call dlgama(alphan, gam2)
alphan = alpha(jnew) + nj2
call dlgama(alphan, gam3)
alphan = alpha(jold) + nj2 - njmove
call dlgama(alphan, gam4)
fggivkln = fggivkl + gam1 - gam2 + gam3 - gam4

return
end



SUBROUTINE updateg2(KMAX,NMAX,XSMAX,dxsm,njmove,gmove,jold,jnew,
A                   pos,g,nj,idg,idgn,nj1,nj2,xsumm,xsumm1,xsumm2,
B                   fggivkl,fxgivkgl,fglog,fggivkln,fxgivkgn,
C                   fgnlog)
implicit none
integer KMAX,NMAX,XSMAX
integer dxsm,njmove,gmove(NMAX),jold,jnew
integer g(NMAX),nj(KMAX),idg,idgn,nj1,nj2,pos(KMAX)
double precision xsumm(XSMAX,KMAX),xsumm1(XSMAX),xsumm2(XSMAX)
double precision fggivkl,fggivkln,fxgivkgl,fxgivkgn,fglog,fgnlog

integer i,pold,pnew,count

pold=pos(jold)
pnew=pos(jnew)

do count=1,njmove
   i = gmove(count)
   g(i) = jnew
enddo
idg=idgn
nj(pold)=nj1
nj(pnew)=nj2
do i=1,dxsm
   xsumm(i,pold)=xsumm1(i)
   xsumm(i,pnew)=xsumm2(i)
enddo

fggivkl=fggivkln
fxgivkgl=fxgivkgn
fglog=fgnlog

return
end



SUBROUTINE metrstep3(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,dxsm,g,
A                    pos,invpos,x,nj,idg,fggivkl,fxgivkgl,fglog,
B                    xsumm,const,alpha,alpha0,phi,mtrprop,mtraccp)
implicit none
integer NMAX,KMAX,HYPMAX,XSMAX
```

```
integer n,k,dxsm
integer g(NMAX),nj(KMAX),idg,pos(KMAX),invpos(KMAX)
integer mtrprop(6),mtraccp(6)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX)
double precision fggivkl,fxgivkgl,fglog,lfk(KMAX)
double precision xsumm(XSMAX,KMAX),const(KMAX),phi(HYPMAX,KMAX)
character fam

integer p1,p2,n12,j,j1,j2,nj1,nj2,njA,njB,i,ii,indi
integer g1(NMAX),ind(NMAX)
double precision uni01kis
double precision lp1,lp2,lmax,lprcand,lprcurr,laccprb
double precision lprob1,lprob2,lprob12,fggivkl1,fxgivkgl1
double precision fg1log,fxgivkgl2,fxgivkglA,fxgivkglB
double precision xsumm1(XSMAX),xsumm2(XSMAX)
double precision xsummA(XSMAX),xsummB(XSMAX)

if(k.eq.1) return

p1 = int(k*UNI01KIS()) + 1
 10   continue
p2 = int(k*UNI01KIS()) + 1
if(p2.eq.p1) go to 10

n12 = nj(p1) + nj(p2)
if(n12.eq.0) return

mtrprop(4) = mtrprop(4) + 1
j1 = invpos(p1)
j2 = invpos(p2)
call zeromet(XSMAX,dxsm,nj1,nj2,xsumm1,xsumm2)

C ind containes indexes of obs in components j1 and j2
ii=0
do i=1,n
   if(g(i).eq.j1.or.g(i).eq.j2) then
      ii = ii + 1
      ind(ii) = i
   endif
enddo

C Randomly permute the entries in ind
ii=n12
20   continue
i = int(ii * UNI01KIS()) + 1
indi = ind(i)
ind(i) = ind(ii)
ind(ii) = indi
ii = ii - 1
if(ii.gt.1) go to 20

lprcand = 0.0d-00
lprcurr = 0.0d-00
do ii=1,n12
   i = ind(ii)
   j=j1
   call copysumm(XSMAX,dxsm,nj1,nj2,xsumm1,xsumm2,njA,njB,
   A             xsummA,xsummB)
   call summetr(NMAX,XSMAX,fam,i,j,j1,njA,njB,x,xsummA,xsummB)
   call fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
   A             j1,j2,p1,p2,njA,njB,xsummA,xsummB,
   B             fxgivkgl,fxgivkgl1)
   j=j2
   call copysumm(XSMAX,dxsm,nj1,nj2,xsumm1,xsumm2,njA,njB,
   A             xsummA,xsummB)
   call summetr(NMAX,XSMAX,fam,i,j,j1,njA,njB,x,xsummA,xsummB)
   call fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
   A             j1,j2,p1,p2,njA,njB,xsummA,xsummB,
   B             fxgivkgl,fxgivkgl2)
   lp1 = dlog(alpha(j1) + dfloat(nj1)) + fxgivkgl1
   lp2 = dlog(alpha(j2) + dfloat(nj2)) + fxgivkgl2

C lp1, lp2 are such that the prob that observation i is proposed
C to be allocated to comp j1 is  1 / [ 1 + exp(lp2 - lp1) ]

   lmax = lp1
   if(lmax.lt.lp2) lmax=lp2
   lprob12 = dlog(dexp(lp1-lmax) + dexp(lp2-lmax))
   lprob1 = lp1 - lmax - lprob12
   lprob2 = lp2 - lmax - lprob12
   if(g(i).eq.j1) then
      lprcurr = lprcurr + lprob1
   else
      lprcurr = lprcurr + lprob2
   endif
   if(dlog(UNI01KIS()).lt.lprob1) then
      j=j1
      lprcand = lprcand + lprob1
   else
      j=j2
      lprcand = lprcand + lprob2
   endif
   g1(i) = j
   call summetr(NMAX,XSMAX,fam,i,j,j1,nj1,nj2,x,xsumm1,xsumm2)
enddo

call fggivkmet(KMAX,k,alpha,alpha0,j1,j2,p1,p2,nj1,nj2,n,n,
A              nj,fggivkl,fggivkl1)
call fxgivkg1(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A             j1,j2,p1,p2,nj1,nj2,xsumm1,xsumm2,
B             fxgivkgl,fxgivkgl1)
laccprb = fxgivkgl1 - fxgivkgl + fggivkl1 - fggivkl
A         + lprcurr - lprcand

if(dlog(UNI01KIS()).lt.laccprb) then
   mtraccp(4) = mtraccp(4) + 1
   fg1log = lfk(k) + fggivkl1 + fxgivkgl1
   call updatmet(NMAX,KMAX,XSMAX,dxsm,n,j1,j2,p1,p2,g,g1,nj,
   A             nj1,nj2,idg,xsumm,xsumm1,xsumm2,fggivkl,
   B             fxgivkgl,fglog,fggivkl1,fxgivkgl1,fg1log)
endif

return
end



SUBROUTINE copysumm(XSMAX,dxsm,nj1,nj2,xsumm1,xsumm2,njA,njB,
A                   xsummA,xsummB)
implicit none
integer XSMAX
integer dxsm,nj1,nj2,njA,njB
double precision xsumm1(XSMAX),xsumm2(XSMAX)
double precision xsummA(XSMAX),xsummB(XSMAX)

integer m

njA = nj1
njB = nj2
do m=1,dxsm
   xsummA(m) = xsumm1(m)
   xsummB(m) = xsumm2(m)
enddo

return
end



SUBROUTINE ejtabs(KMAX,NMAX,HYPMAX,XSMAX,km,n,k,fam,lfk,dxsm,g,
A                 pos,invpos,firstav,x,nj,idg,fggivkl,fxgivkgl,
B                 fglog,xsumm,const,alpha,alpha0,phi,indsym,
C                 ejectp,mtrprop,mtraccp)
implicit none
integer NMAX,KMAX,HYPMAX,XSMAX
integer km,n,k,dxsm,idg,indsym
integer pos(KMAX),invpos(KMAX),firstav,g(NMAX),nj(KMAX)
integer mtrprop(6),mtraccp(6)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX),lfk(KMAX)
double precision fggivkl,fxgivkgl,fglog
double precision xsumm(XSMAX,KMAX),const(KMAX)
double precision phi(HYPMAX,KMAX),ejectp(KMAX)
character fam

double precision p0
double precision uni01kis

p0 = 0.2d-00

if(UNI01KIS() .le. ejectp(k)) then

   call eject(KMAX,NMAX,HYPMAX,XSMAX,km,n,k,fam,lfk,dxsm,g,pos,
   A          invpos,firstav,x,nj,idg,fggivkl,fxgivkgl,fglog,
   B          xsumm,const,alpha,alpha0,phi,ejectp,p0,indsym,
   C          mtrprop,mtraccp)
else
```

```
   call absorb(KMAX,NMAX,HYPMAX,XSMAX,km,n,k,fam,lfk,dxsm,g,pos,
   A               invpos,firstav,x,nj,idg,fggivkl,fxgivkgl,fglog,
   B               xsumm,const,alpha,alpha0,phi,ejectp,p0,indsym,
   C               mtrprop,mtraccp)
endif

return
end



SUBROUTINE getpejt(n, p0, pejt, a)
implicit none
integer n
double precision p0, pejt, a

integer nn,ind
double precision gamdev
double precision logn, rem
double precision a01(12), a02(12), a03(12), a04(12), a05(12)
data a01/58.25065d-00, 2.4572173d-00, 0.9690759d-00,
A       0.6354756d-00, 0.4816089d-00, 0.3908933d-00,
B       0.3302802d-00, 0.2865916d-00, 0.2534592d-00,
C       0.2273947d-00, 0.2063139d-00, 0.1888896d-00/
data a02/11.0258d-00, 1.2138196d-00, 0.6033860d-00,
A       0.4170041d-00, 0.3230759d-00, 0.2653151d-00,
B       0.2257773d-00, 0.1968376d-00, 0.1746576d-00,
C       0.1570764d-00, 0.1427763d-00, 0.1309047d-00/
data a03/8.922985d-00, 0.7612973d-00, 0.4211680d-00,
A       0.2999519d-00, 0.2355229d-00, 0.1948357d-00,
B       0.1665482d-00, 0.1456349d-00, 0.1294961d-00,
C       0.1166401d-00, 0.1061447d-00, 0.0974070d-00/
data a04/3.1721607d-00, 0.5153639d-00, 0.3048555d-00,
A       0.2217515d-00, 0.1758312d-00, 0.1462478d-00,
B       0.1254355d-00, 0.1099308d-00, 0.0979028d-00,
C       0.0882851d-00, 0.0804112d-00, 0.0738418d-00/
data a05/1.2920201d-00, 0.3572630d-00, 0.2216664d-00,
A       0.1638918d-00, 0.1309661d-00, 0.1094074d-00,
B       0.0940931d-00, 0.0826123d-00, 0.0736672d-00,
C       0.0664924d-00, 0.0606049d-00, 0.0556840d-00/

C a01(i) contains a corresponding to p0 = 0.1 for n=exp(i)
C            similarly for a02, ..., a05
C computed from calls to the Splus function findabisct()
C log(a01(1)) was determined by linear extrapolation through
C the points (log(5), log(8.460785)) and (2, log(a01(2)))
C log(a02(1)) was determined by linear extrapolation through
C the points (log(4), log(4.701553)) and (2, log(a02(2)))
C log(a03(1)) was determined by linear extrapolation through
C the points (log(3), log(6.999999)) and (2, log(a03(2)))

if(n.lt.3) then
   nn = 3
else
   nn = n
endif
logn = dlog(dfloat(nn))
ind = int(logn)
rem = logn - dfloat(ind)

if(p0.eq.0.1d-00) then
   if(ind.lt.12) then
      a = dexp( dlog(a01(ind)) +
      A         rem * (dlog(a01(ind+1)) - log(a01(ind))) )
   else
      a = dexp( dlog(a01(12)) +
      A         (logn-12.d-00)*(dlog(a01(12))-log(a01(11))) )
   endif
   go to 10
endif

if(p0.eq.0.2d-00) then
   if(ind.lt.12) then
      a = dexp( dlog(a02(ind)) +
      A         rem * (dlog(a02(ind+1)) - log(a02(ind))) )
   else
      a = dexp( dlog(a02(12)) +
      A         (logn-12.d-00)*(dlog(a02(12))-log(a02(11))) )
   endif
   go to 10
endif

if(p0.eq.0.3d-00) then
   if(ind.lt.12) then
      a = dexp( dlog(a03(ind)) +
      A         rem * (dlog(a03(ind+1)) - log(a03(ind))) )
   else
      a = dexp( dlog(a03(12)) +
      A         (logn-12.d-00)*(dlog(a03(12))-log(a03(11))) )
   endif
   go to 10
endif

if(p0.eq.0.4d-00) then
   if(ind.lt.12) then
      a = dexp( dlog(a04(ind)) +
      A         rem * (dlog(a04(ind+1)) - log(a04(ind))) )
   else
      a = dexp( dlog(a04(12)) +
      A         (logn-12.d-00)*(dlog(a04(12))-log(a04(11))) )
   endif
   go to 10
endif

if(p0.eq.0.5d-00) then
   if(ind.lt.12) then
      a = dexp( dlog(a05(ind)) +
      A         rem * (dlog(a05(ind+1)) - log(a05(ind))) )
   else
      a = dexp( dlog(a05(12)) +
      A         (logn - 12.d-00) * (dlog(a05(12)) - log(a05(11))) )
   endif
   go to 10
endif

write(6,*) "wrong p0 input for getpejt"
stop

10   continue
pejt = 1.0d-00 / (1.0d-00 + gamdev(a)/gamdev(a))

return
end



SUBROUTINE eject(KMAX,NMAX,HYPMAX,XSMAX,km,n,k,fam,lfk,dxsm,g,pos,
A                invpos,firstav,x,nj,idg,fggivkl,fxgivkgl,fglog,
B                xsumm,const,alpha,alpha0,phi,ejectp,p0,indsym,
C                mtrprop,mtraccp)
implicit none
integer NMAX,KMAX,HYPMAX,XSMAX
integer km,n,k,dxsm
integer pos(KMAX),invpos(KMAX),firstav,g(NMAX),nj(KMAX),idg,indsym
integer mtrprop(6),mtraccp(6)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX),lfk(KMAX)
double precision fggivkl,fxgivkgl,fglog
double precision xsumm(XSMAX,KMAX),const(KMAX),phi(HYPMAX,KMAX)
double precision ejectp(KMAX),p0
character fam

integer ejectingc,ejectedc,ejectingp,ejectedp
integer i,idgn,njted,njting,igchng(NMAX)
double precision uni01kis
double precision xsumting(XSMAX),xsumted(XSMAX)
double precision fggivkl1,fxgivkgl1,fg1log
double precision lpalloc,laccprob,lfggivkrat,lfxgivkgrat
double precision pejt,a

mtrprop(5) = mtrprop(5) + 1

C Select ejecting component with prob 1/k

ejectingc = invpos(int(k*UNI01KIS())+1)
ejectingp = pos(ejectingc)

C Ejected component goes to position (k+1) in pos

ejectedc = firstav
if(indsym.eq.0) then
   ejectedp = ejectedc
 else
   ejectedp = int((k+1)*UNI01KIS())+1
endif
```

```
C Create igchng which holds indices of ejected comp

call getpejt(nj(ejectingp), p0, pejt, a)
njted = 0
do i=1,n
   if(g(i).eq.ejectingc.and.UNI01KIS().lt.pejt) then
      njted = njted + 1
      igchng(njted) = i
   endif
enddo
njting = nj(ejectingp) - njted
if(njted.ne.0.and.njting.ne.0) then
   idgn = idg+1
else
   idgn = idg
endif

C Update summaries xsumm

call summejt(KMAX,NMAX,XSMAX,fam,dxsm,n,x,g,igchng,njted,
A            njting,ejectingc,ejectingp,xsumm,xsumting,xsumted)

C Accept/Reject

call fggivkejab(KMAX,n,k,alpha,alpha0,njting,njted,ejectingc,
A               ejectedc,lfggivkrat)

call fxgivkgejt(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A                ejectingc,ejectedc,ejectingp,njting,njted,
B                xsumting,xsumted,lfxgivkgrat)

call comppalloc(njted, njting, a, lpalloc)

laccprob = lfk(k+1) - lfk(k) + lfggivkrat + lfxgivkgrat
A          - lpalloc
B          + dlog(1.0d-00 - ejectp(k+1)) - dlog(ejectp(k))

if(laccprob.gt.dlog(UNI01KIS())) then
   fggivkl1 = fggivkl + lfggivkrat
   fxgivkgl1 = fxgivkgl + lfxgivkgrat
   fg1log = lfk(k+1) + fggivkl1 + fxgivkgl1
   call updatejt(KMAX,NMAX,XSMAX,km,k,dxsm,g,igchng,pos,invpos,
   A             firstav,ejectingc,ejectedc,ejectedp,nj,njting,
   B             njted,idg,idgn,xsumm,xsumting,xsumted,fggivkl,
   C             fxgivkgl,fglog,fggivkl1,fxgivkgl1,fg1log)
   mtraccp(5) = mtraccp(5) + 1
endif

return
end



SUBROUTINE summejt(KMAX,NMAX,XSMAX,fam,dxsm,n,x,g,igchng,
A                  njted,njting,ejectingc,ejectingp,xsumm,
B                  xsumting,xsumted)
implicit none
integer NMAX,KMAX,XSMAX
integer n,dxsm,njted,njting,ejectingc,ejectingp
integer g(NMAX),igchng(NMAX)
double precision x(NMAX),xsumm(XSMAX,KMAX)
double precision xsumting(XSMAX),xsumted(XSMAX)
character fam

if(fam.eq."N") then
   call summejtN(KMAX,NMAX,XSMAX,dxsm,x,igchng,njted,
   A             ejectingp,xsumm,xsumting,xsumted)
   return
endif
write(6,*) "Wrong family in Sub. summejt"
stop

end



SUBROUTINE fggivkejab(KMAX,n,k,alpha,alpha0,njting,njted,
A                     ejectingc,ejectedc,term)
implicit none
integer KMAX,n,k
integer njting,njted,ejectingc,ejectedc
double precision alpha(KMAX),alpha0(KMAX)
double precision term

double precision salphk,salphk1,salphkn,salphk1n
double precision alphnj,alphnjing,alphnjted,alphted
double precision gam1,gam2,gam3,gam4,gam5,gam6,gam7,gam8

salphk = alpha0(k)
salphk1 = alpha0(k+1)
salphkn = salphk + dfloat(n)
salphk1n = salphk1 + dfloat(n)
alphnj = alpha(ejectingc) + njting + njted
alphnjing = alphnj - njted
alphnjted = alpha(ejectedc) + njted
alphted = alpha(ejectedc)

call dlgama(salphk,gam1)
call dlgama(salphk1,gam2)
call dlgama(salphkn,gam3)
call dlgama(salphk1n,gam4)
call dlgama(alphnj,gam5)
call dlgama(alphnjing,gam6)
call dlgama(alphnjted,gam7)
call dlgama(alphted,gam8)

term = (gam2+gam3+gam6+gam7) - (gam1+gam4+gam5+gam8)

return
end



SUBROUTINE fxgivkgejt(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A                     ejectingc,ejectedc,ejectingp,njting,njted,
B                     xsumting,xsumted,lfxgivkgrat)
implicit none
integer KMAX,HYPMAX,XSMAX
integer nj(KMAX),njting,njted
integer ejectingc,ejectedc,ejectingp
double precision xsumm(XSMAX,KMAX),const(KMAX),xsumting(XSMAX)
double precision xsumted(XSMAX),phi(HYPMAX,KMAX)
double precision lfxgivkgrat
character fam

if(fam.eq."N") then
   call fxgivkgejtN(KMAX,HYPMAX,XSMAX,nj,xsumm,const,phi,
   A                ejectingc,ejectedc,ejectingp,njting,njted,
   B                xsumting,xsumted,lfxgivkgrat)
   return
endif
write(6,*) "Wrong family in Sub. fxgivkgejt"
stop

end



SUBROUTINE comppalloc(njted, njting, a, lpalloc)
implicit none
integer njted,njting
double precision a,lpalloc

double precision anjted,anjting,twoan,twoa
double precision gam1,gam2,gam3,gam4,gam5

anjted = a + dfloat(njted)
anjting = a + dfloat(njting)
twoan = anjted + anjting
twoa = 2.0d-00 * a

call dlgama(a, gam1)
call dlgama(twoa, gam2)
call dlgama(anjted, gam3)
call dlgama(anjting, gam4)
call dlgama(twoan, gam5)

lpalloc = gam2 + gam3 + gam4 - gam1 - gam1 - gam5

return
end



SUBROUTINE updatejt(KMAX,NMAX,XSMAX,km,k,dxsm,g,igchng,pos,invpos,
A                   firstav,ejectingc,ejectedc,ejectedp,nj,njting,
```

```
B                  njted,idg,idgn,xsumm,xsumting,xsumted,fggivkl,
C                  fxgivkgl,fglog,fggivkl1,fxgivkgl1,fg1log)
implicit none
integer KMAX,NMAX,XSMAX
integer km,k,dxsm,idg,idgn
integer pos(KMAX),invpos(KMAX),firstav,g(NMAX),igchng(NMAX)
integer ejectingc,ejectedc,ejectedp,nj(KMAX),njting,njted
double precision xsumm(XSMAX,KMAX),xsumting(XSMAX)
double precision xsumted(XSMAX)
double precision fggivkl,fggivkl1,fxgivkgl,fxgivkgl1
double precision fglog,fg1log

integer i,m,newfirst

k = k+1
idg = idgn
do i=1,njted
   g(igchng(i)) = ejectedc
enddo

if(ejectedp.eq.k) then
   pos(ejectedc) = ejectedp
   invpos(ejectedp) = ejectedc
else
   pos(ejectedc) = ejectedp
   pos(invpos(ejectedp)) = k
   invpos(k) = invpos(ejectedp)
   invpos(ejectedp) = ejectedc
endif

nj(k)=nj(pos(ejectedc))
nj(pos(ejectingc))=njting
nj(pos(ejectedc))=njted
do m=1,dxsm
   xsumm(m,k)=xsumm(m,pos(ejectedc))
   xsumm(m,pos(ejectingc))=xsumting(m)
   xsumm(m,pos(ejectedc))=xsumted(m)
enddo

if(k.eq.km) then
   firstav =  1000000
else
   do i=firstav+1,km
      if(pos(i).eq.0) then
         newfirst=i
         go to 10
      endif
   enddo
10      continue
   firstav=newfirst
endif

fggivkl=fggivkl1
fxgivkgl=fxgivkgl1
fglog = fg1log

return
end



SUBROUTINE absorb(KMAX,NMAX,HYPMAX,XSMAX,km,n,k,fam,lfk,dxsm,g,
A                 pos,invpos,firstav,x,nj,idg,fggivkl,fxgivkgl,
B                 fglog,xsumm,const,alpha,alpha0,phi,ejectp,p0,
C                 indsym,mtrprop,mtraccp)
implicit none
integer NMAX,KMAX,HYPMAX,XSMAX
integer km,n,k,dxsm,indsym
integer pos(KMAX),invpos(KMAX),firstav,g(NMAX),nj(KMAX),idg
integer mtrprop(6),mtraccp(6)
double precision x(NMAX),alpha(KMAX),alpha0(KMAX),lfk(KMAX)
double precision fggivkl,fxgivkgl,fglog
double precision xsumm(XSMAX,KMAX),const(KMAX),phi(HYPMAX,KMAX)
double precision ejectp(KMAX),p0
character fam

integer absorbingc,absorbedc,absorbingp,absorbedp
integer njbing,njbed,njabs,idgn
double precision uni01kis
double precision fggivkl1,fxgivkgl1,fg1log,lpalloc,laccprob
double precision xsumabs(XSMAX)
double precision lfggivkrat,lfxgivkgrat
double precision pejt,a
mtrprop(6) = mtrprop(6) + 1

C Select absorbing and absorbed component

if(indsym.eq.0) then
   absorbingc = invpos(int((k-1)*UNI01KIS())+1)
   absorbedc = k
else
   absorbingc = invpos(int(k*UNI01KIS())+1)
10      continue
   absorbedc = invpos(int(k*UNI01KIS())+1)
   if (absorbedc.eq.absorbingc) goto 10
endif

absorbingp = pos(absorbingc)
absorbedp = pos(absorbedc)
njbing = nj(absorbingp)
njbed = nj(absorbedp)
njabs = njbing + njbed

call getpejt(njabs, p0, pejt, a)

idgn = idg
if(nj(absorbingp).gt.0.and.nj(absorbedp).gt.0) idgn=idg-1

C Update summaries xsumm

call summabs(KMAX,NMAX,XSMAX,fam,absorbingp,absorbedp,njbing,
A            njbed,xsumm,xsumabs)

C Accept/Reject

call fggivkejab(KMAX,n,k-1,alpha,alpha0,njbing,njbed,absorbingc,
A               absorbedc,lfggivkrat)

call fxgivkgabs(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
A               absorbingc,absorbedc,absorbingp,absorbedp,
B               njabs,xsumabs,lfxgivkgrat)

call comppalloc(njbed, njbing, a, lpalloc)

laccprob = lfk(k-1) - lfk(k) - lfggivkrat - lfxgivkgrat
A          + lpalloc
B          - dlog(1.0d-00 - ejectp(k)) + dlog(ejectp(k-1))

if(laccprob.gt.dlog(UNI01KIS())) then
   fggivkl1 = fggivkl - lfggivkrat
   fxgivkgl1 = fxgivkgl - lfxgivkgrat
   fg1log = lfk(k-1) + fggivkl1 + fxgivkgl1
   call updatabs(KMAX,NMAX,XSMAX,indsym,n,k,dxsm,g,pos,invpos,
   A             firstav,absorbingc,absorbedc,nj,njabs,idg,idgn,
   B             xsumm,xsumabs,fggivkl,fxgivkgl,fglog,fggivkl1,
   C             fxgivkgl1,fg1log,km)
   mtraccp(6) = mtraccp(6) + 1
endif

return
end



SUBROUTINE summabs(KMAX,NMAX,XSMAX,fam,absorbingp,absorbedp,
A                  njbing,njbed,xsumm,xsumabs)
implicit none
integer NMAX,KMAX,XSMAX
integer absorbingp,absorbedp,njbing,njbed
double precision xsumm(XSMAX,KMAX),xsumabs(XSMAX)
character fam

if(fam.eq."N") then
   call summabsN(KMAX,NMAX,XSMAX,absorbingp,absorbedp,xsumm,
   A             xsumabs)
   return
endif
write(6,*) "Wrong family in Sub. summabs"
stop

end



SUBROUTINE fxgivkgabs(KMAX,HYPMAX,XSMAX,fam,nj,xsumm,const,phi,
```

```
A                    absorbingc,absorbedc,absorbingp,absorbedp,
B                    njabs,xsumabs,lfxgivkgrat)
implicit none
integer KMAX,HYPMAX,XSMAX
integer nj(KMAX),njabs
integer absorbingc,absorbedc,absorbingp,absorbedp
double precision xsumm(XSMAX,KMAX),const(KMAX),xsumabs(XSMAX)
double precision phi(HYPMAX,KMAX)
double precision lfxgivkgrat
character fam

if(fam.eq."N") then
   call fxgivkgabsN(KMAX,HYPMAX,XSMAX,nj,xsumm,const,phi,
   A                absorbingc,absorbedc,absorbingp,absorbedp,
   B                njabs,xsumabs,lfxgivkgrat)
   return
endif
write(6,*) "Wrong family in Sub. fxgivkgabs"
stop

end



SUBROUTINE updatabs(KMAX,NMAX,XSMAX,indsym,n,k,dxsm,g,pos,
A                   invpos,firstav,absorbingc,absorbedc,nj,njabs,
B                   idg,idgn,xsumm,xsumabs,fggivkl,fxgivkgl,
C                   fglog,fggivkl1,fxgivkgl1,fg1log,km)
implicit none
integer KMAX,NMAX,XSMAX
integer indsym,n,k,dxsm,idg,idgn,km
integer pos(KMAX),invpos(KMAX),firstav,g(NMAX)
integer absorbingc,absorbedc,nj(KMAX),njabs
double precision xsumm(XSMAX,KMAX),xsumabs(XSMAX)
double precision fggivkl,fggivkl1,fxgivkgl,fxgivkgl1
double precision fglog,fg1log

integer i,m

k = k-1
idg = idgn

do i=1,n
   if(g(i).eq.absorbedc) g(i) = absorbingc
enddo

nj(pos(absorbingc))=njabs
nj(pos(absorbedc))=nj(k+1)
nj(k+1)=0

do m=1,dxsm
   xsumm(m,pos(absorbingc))=xsumabs(m)
   xsumm(m,pos(absorbedc))=xsumm(m,k+1)
   xsumm(m,k+1)=0.0d-00
enddo

pos(invpos(k+1)) = pos(absorbedc)
invpos(pos(absorbedc)) = invpos(k+1)
pos(absorbedc) = 0
invpos(k+1) = 0
if(absorbedc.lt.firstav) firstav=absorbedc

fggivkl=fggivkl1
fxgivkgl=fxgivkgl1
fglog = fg1log

return
end



SUBROUTINE metrolab(KMAX,NMAX,HYPMAX,XSMAX,n,k,fam,lfk,g,pos,
A                   invpos,nj,fggivkl,fxgivkgl,fglog,xsumm,
B                   const,alpha,phi,indsym)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer n,k,indsym
integer g(NMAX),nj(KMAX),pos(KMAX),invpos(KMAX)
double precision fglog,fggivkl,fxgivkgl
double precision xsumm(XSMAX,KMAX),const(KMAX),alpha(KMAX)
double precision phi(HYPMAX,KMAX),lfk(KMAX)
character fam
integer i,p1,p2,j1,j2,nj1,nj2
double precision alphan,gam1,gam2,gam3,gam4
double precision laccprob,fgnlog,fggivkln,fxgivkgn,uni01kis

C Metropolis move on the labels
C Only attempted if indsym=0, i.e. asymmetric case

if(indsym.ne.0) return
if(k.eq.1) return

p1 = int(k*UNI01KIS()) + 1
10   continue
p2 = int(k*UNI01KIS()) + 1
if(p2.eq.p1) go to 10

j1 = invpos(p1)
j2 = invpos(p2)
nj1 = nj(p1)
nj2 = nj(p2)

alphan = alpha(j1) + nj1
call dlgama(alphan, gam1)
alphan = alpha(j2) + nj2
call dlgama(alphan, gam2)
alphan = alpha(j1) + nj2
call dlgama(alphan, gam3)
alphan = alpha(j2) + nj1
call dlgama(alphan, gam4)
fggivkln = fggivkl - gam1 - gam2 + gam3 + gam4

call fxgivkgmlab(KMAX,HYPMAX,XSMAX,fam,j1,j2,p1,p2,nj1,nj2,phi,
A                xsumm,const,fxgivkgl,fxgivkgn)

fgnlog = lfk(k) + fggivkln + fxgivkgn
laccprob =  fgnlog - fglog
if(laccprob.gt.dlog(UNI01KIS())) then
   pos(j1) = p2
   pos(j2) = p1
   invpos(p1) = j2
   invpos(p2) = j1
   do i=1,n
      if(g(i).eq.j1) then
         g(i)=j2
      else
         if(g(i).eq.j2) g(i)=j1
      endif
   enddo
   fggivkl = fggivkln
   fxgivkgl = fxgivkgn
   fglog = fgnlog
endif

return
end



SUBROUTINE fxgivkgmlab(KMAX,HYPMAX,XSMAX,fam,j1,j2,p1,p2,nj1,nj2,
A                      phi,xsumm,const,fxgivkgl,fxgivkgn)
implicit none
integer KMAX,HYPMAX,XSMAX
integer j1,j2,p1,p2,nj1,nj2
double precision phi(HYPMAX,KMAX),xsumm(XSMAX,KMAX)
double precision const(KMAX),fxgivkgl,fxgivkgn
character fam

if(fam.eq."N") then
   call fxgivkgmlabN(KMAX,HYPMAX,XSMAX,j1,j2,p1,p2,nj1,nj2,
   A                 phi,xsumm,const,fxgivkgl,fxgivkgn)
   return
endif
write(6,*) "Wrong family in Sub. fxgivkgmlab"
stop

end



SUBROUTINE writekgout(KMAX,NMAX,k,n,g,pos,nj,idg,indsym,ouk,oug)
implicit none
integer KMAX,NMAX
integer k,n,g(NMAX),pos(KMAX),nj(KMAX),idg,indsym,ouk,oug
```

```
integer i,outg(NMAX)
character gchar(NMAX)

write(ouk,1) k, idg

if(indsym.eq.1) then
   call remgaps(KMAX,NMAX,k,n,g,pos,nj,idg,outg)
   call gint2char(NMAX,n,outg,gchar)
else
   call gint2char(NMAX,n,g,gchar)
endif

write(oug,2) (gchar(i),i=1,n)

1    format(2i3)
2    format(1000000a1)

return
end



SUBROUTINE remgaps(KMAX,NMAX,k,n,g,pos,nj,idg,outg)
implicit none
integer KMAX,NMAX
integer k,n,g(NMAX),pos(KMAX),nj(KMAX),idg,outg(NMAX)

integer i,j,nempt

do i=1,n
   outg(i) = pos(g(i))
enddo
if(k.gt.idg) then
   nempt=0
   do j=1,k
      if(nj(j).eq.0) then
         nempt = nempt+1
      else
         do i=1,n
            if(outg(i).eq.j) outg(i) = outg(i) - nempt
         enddo
      endif
    enddo
endif

return
end



SUBROUTINE relabgnat(KMAX,NMAX,k,n,g)
implicit none
integer KMAX,NMAX
integer k,n,g(NMAX)

C Changes the labels in the vector G so that
C        they are in the natural order

integer j,ind(KMAX),ngrp,i,ig

do 10 j=1,k
   ind(j)=0
10   continue

ngrp = 0

do 20 i=1,n
   ig=g(i)
   if(ind(ig).eq.0) then
      ngrp=ngrp+1
      ind(ig)=ngrp
   endif
20   continue

do 30 i=1,n
   ig=g(i)
   g(i)=ind(ig)
30   continue

return
end
```

```
SUBROUTINE writehyp(KMAX,HYPMAX,fam,phi,ouhyp)
implicit none
integer KMAX,HYPMAX
integer ouhyp
double precision phi(HYPMAX,KMAX)
character fam

if(fam.eq."N") then
   call writehypN(KMAX,HYPMAX,phi,ouhyp)
   return
endif
write(6,*) "Wrong family in Sub. writehyp"
stop

end



SUBROUTINE SUMMRYCHK(KMAX,NMAX,XSMAX,km,n,dxsm,k,fam,g,pos,
A                    invpos,x,nj,idg,xsumm)
implicit none
integer KMAX,NMAX,XSMAX
integer n,k,idg,km,dxsm
integer g(NMAX),nj(KMAX),pos(KMAX),invpos(KMAX)
double precision x(NMAX),xsumm(XSMAX,KMAX)
character fam

integer idgX,njX(KMAX),j,noteq,ii
double precision xsummX(XSMAX,KMAX),eps
parameter (eps=1.0d-07)

call selecxg(KMAX,NMAX,XSMAX,km,n,k,fam,g,pos,x,njX,xsummX,idgX)
noteq = 0
if(idg.ne.idgX) noteq = noteq + 1
do ii=1,k
   if(nj(ii).ne.njX(ii)) noteq = noteq + 1
   do j=1,dxsm
      if(dabs(xsumm(j,ii)-xsummX(j,ii)).gt.eps) noteq = noteq +1
   enddo
enddo
if(noteq.ne.0) then
   write(6,*) "k=",k
   write(6,*) "idg=",idg
   write(6,*) "g=", (g(ii),ii=1,n)
   write(6,*) "nj=", (nj(ii),ii=1,k)
   write(6,*) "pos=", (pos(ii),ii=1,km)
   write(6,*) "invpos=", (invpos(ii),ii=1,km)
   do j=1,dxsm
      write(6,*) "xsumm(",j,",)=", (xsumm(j,ii),ii=1,k)
   enddo
   write(6,*) "noteq=",noteq
   write(6,*) "idgX=",idgX
   write(6,*) "njX=", (njX(ii),ii=1,k)
   do j=1,dxsm
      write(6,*) "xsummX(",j,",)=", (xsummX(j,ii),ii=1,k)
   enddo
   stop
endif

return
end



SUBROUTINE metrohypN(KMAX,NMAX,HYPMAX,XSMAX,i,burnin,km,n,x,k,
A                    invpos,nj,xsumm,const,phi,fxgivkgl,fglog)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer i,burnin,km,n,k,invpos(KMAX),nj(KMAX)
double precision x(NMAX),xsumm(XSMAX,KMAX),const(KMAX)
double precision phi(HYPMAX,KMAX),fxgivkgl,fglog

C Only for tau and delta (assumed the SAME for all components)
C priors: 1/(1+tau) ~ Un(0,1),   delta ~ Un(0, ubvarwit) independently
C         ubvarwit is  equal to Var(data) * (gamma-1)

integer j
save ubvarwit
double precision uni01kis,varx,ubvarwit,reptau1,reptau1prp
double precision tauprp,deltaprp,savetau,savedelta,fxgivkgl1
double precision reptau1min,reptau1max,prpmax,prpmin
double precision deltamin,deltamax,proprat,logproprat
```

```
double precision saveconst(KMAX)
double precision logtargrat

if(i.eq.-burnin) then
   call compvarxN(NMAX,n,x,varx)
   if(phi(3,1).ge.2.0d-00) then
      ubvarwit = 1.d-00 * varx * (phi(3,1) - 1.0d-00)
    else
      ubvarwit = 1.d-00 * varx
    endif
endif

reptau1 = 1.0d-00 / (1.0d-00 + phi(2,1))
reptau1min = max(reptau1 - 0.01d-00, 0.0d-00)
reptau1max = min(reptau1 + 0.01d-00, 1.0d-00)
if(UNI01KIS().lt.0.5d-00) then
   reptau1prp = reptau1 - UNI01KIS() * (reptau1 - reptau1min)
   prpmax = min(reptau1prp + 0.01d-00, 1.0d-00)
   proprat = (reptau1 - reptau1min)/(prpmax - reptau1prp)
else
   reptau1prp = reptau1 + UNI01KIS() * (reptau1max - reptau1)
   prpmin = max(reptau1prp - 0.01d-00, 0.0d-00)
   proprat = (reptau1max - reptau1)/(reptau1prp - prpmin)
endif
logproprat = dlog(proprat)

deltamin = max(phi(4,1) - ubvarwit * 0.01d-00, 0.0d-00)
deltamax = min(phi(4,1) + ubvarwit * 0.01d-00, ubvarwit)
if(UNI01KIS().lt.0.5d-00) then
   deltaprp = phi(4,1) - UNI01KIS() * (phi(4,1) - deltamin)
   prpmax = min(deltaprp + ubvarwit * 0.01d-00, ubvarwit)
   proprat = (phi(4,1) - deltamin)/(prpmax - deltaprp)
else
   deltaprp = phi(4,1) + UNI01KIS() * (deltamax - phi(4,1))
   prpmin = max(deltaprp - ubvarwit * 0.01d-00, 0.0d-00)
   proprat = (deltamax - phi(4,1))/(deltaprp - prpmin)
endif
logproprat = logproprat + dlog(proprat)

tauprp = (1.0d-00 / reptau1prp) - 1.0d-00
savetau = phi(2,1)
savedelta = phi(4,1)
do j=1,km
   saveconst(j)=const(j)
enddo
do j=1,km
   phi(2,j)=tauprp
   phi(4,j)=deltaprp
enddo
call getconstN(KMAX,HYPMAX,km,phi,const)
call fxgivkgN(KMAX,HYPMAX,XSMAX,n,k,invpos,nj,xsumm,const,
A             phi,fxgivkgl1)
logtargrat = fxgivkgl1 - fxgivkgl
if((logtargrat + logproprat) .gt. dlog(UNI01KIS())) then
   fglog=fglog-fxgivkgl+fxgivkgl1
   fxgivkgl=fxgivkgl1
else
   do j=1,km
      phi(2,j)=savetau
      phi(4,j)=savedelta
      const(j)=saveconst(j)
   enddo
endif
10   continue

return
end



SUBROUTINE compvarxN(NMAX,n,x,varx)
implicit none
integer NMAX,n
double precision x(NMAX), varx

integer i
double precision meanx

meanx=0.0d-00
do i=1,n
   meanx = meanx + x(i)
enddo
meanx = meanx / dfloat(n)

varx = 0.0d-00
do i=1,n
   varx = varx + (x(i) - meanx)**2.0d-00
enddo
varx = varx / dfloat(n-1)

return
end



SUBROUTINE getconstN(KMAX,HYPMAX,km,phi,const)
implicit none
integer KMAX,HYPMAX
integer km
double precision phi(HYPMAX,KMAX)
double precision const(KMAX)

integer j
double precision gammaj,gam1

do j=1,km
   gammaj = phi(3,j)
   call dlgama(gammaj,gam1)
   const(j)= -gam1
enddo

do j=1,km
   const(j) = const(j) + phi(3,j)*dlog(2.0d-00*phi(4,j))
enddo

return
end



SUBROUTINE selecxgN(KMAX,NMAX,XSMAX,km,n,k,g,pos,x,nj,xsumm,idg)
implicit none
integer KMAX,NMAX,XSMAX
integer km,k,n
integer g(NMAX),nj(KMAX),pos(KMAX),idg
double precision x(NMAX),xsumm(XSMAX,KMAX)

C x is a n-vector of data, g is a n-vector of memberships
C pos is a vector giving the actual position of component j
C 3 k-vectors and a scalar are returned:
C nj:    nj(p)=card{i|g(i)=j}
C xsumm(1,p):    xsumm(1,p)=sum{ x(i), i in {i|g(i)=j} }
C xsumm(2,p):    xsumm(2,p)=sum{(x(i))**2, i in {i|g(i)=j} }
C                                              j=invpos(p),  p=1,k
C idg:   card{n(p) != 0, p=1,k}

integer ig,i,p

idg=0
do p=1,km
   nj(p)=0
   xsumm(1,p)=0.0d-00
   xsumm(2,p)=0.0d-00
enddo

do i=1,n
   ig=g(i)
   p=pos(ig)
   nj(p)=nj(p)+1
   xsumm(1,p)=xsumm(1,p)+x(i)
   xsumm(2,p)=xsumm(2,p)+x(i)**2
enddo

do p=1,k
   if(nj(p).ne.0) idg=idg+1
enddo

return
end



SUBROUTINE fxgivkgN(KMAX,HYPMAX,XSMAX,n,k,invpos,nj,xsumm,const,
A                   phi,fxgivkgl)
implicit none
integer KMAX,HYPMAX,XSMAX
integer n,k
```

```
integer invpos(KMAX),nj(KMAX)
double precision xsumm(XSMAX,KMAX),fxgivkgl
double precision phi(HYPMAX,KMAX),const(KMAX)

C Computes     log f_(X|K,G)

double precision pi
parameter (pi=0.3141592653589d+01)
integer j,p,njj
double precision term

fxgivkgl= -(dfloat(n)/2.0d-00)*dlog(pi)

do p=1,k
   j=invpos(p)
   njj=nj(p)
   call comptermN(KMAX,HYPMAX,XSMAX,j,p,njj,phi,xsumm,const,
   A              term)
   fxgivkgl = fxgivkgl + term
enddo

return
end



SUBROUTINE comptermN(KMAX,HYPMAX,XSMAX,j,p,njj,phi,xsumm,const,
A                    term)
implicit none
integer KMAX,HYPMAX,XSMAX
integer j,p,njj
double precision phi(HYPMAX,KMAX)
double precision xsumm(XSMAX,KMAX),const(KMAX),term

double precision gamnj2,taunj,gam2,quad
double precision xmean,x2mean

if(njj.gt.0) then
   xmean = xsumm(1,p)/dfloat(njj)
   x2mean = xsumm(2,p) - (xsumm(1,p) ** 2.0d-00) / dfloat(njj)
   if(x2mean.lt.0.0d-00) x2mean = 0.0d-00
   gamnj2=phi(3,j)+dfloat(njj)/2.0d-00
   taunj=phi(2,j)+dfloat(njj)
   call dlgama(gamnj2,gam2)
   quad=x2mean+((xmean-phi(1,j))**2)*phi(2,j)*njj/taunj
   term =   gam2
   A        + const(j)
   B        + 0.5d-00*dlog(phi(2,j)/taunj)
   C        - gamnj2*dlog(2.0d-00*phi(4,j)+quad)

else
   term=0.0d-00
endif

return
end



SUBROUTINE selecxg1N(KMAX,NMAX,HYPMAX,XSMAX,x,igcomp,oldgi,
A                    newgi,pold,pnew,nj,xsumm,idg,idgn,n1,n2,
B                    xsumm1,xsumm2)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer igcomp,oldgi,newgi,pold,pnew
integer nj(KMAX),idg,idgn,n1,n2
double precision x(NMAX),xsumm(XSMAX,KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX)

idgn=idg
if(nj(pold).eq.1) idgn=idgn-1
if(nj(pnew).eq.0) idgn=idgn+1
n1=nj(pold)-1
n2=nj(pnew)+1
if(n1.eq.0) then
   xsumm1(1)=0.0d-00
   xsumm1(2)=0.0d-00
else
   xsumm1(1)=xsumm(1,pold) - x(igcomp)
   xsumm1(2)=xsumm(2,pold) - x(igcomp)**2
endif
xsumm2(1)=xsumm(1,pnew) + x(igcomp)
xsumm2(2)=xsumm(2,pnew) + x(igcomp)**2
return
end



SUBROUTINE fxgivkg1N(KMAX,HYPMAX,XSMAX,nj,xsumm,const,phi,
A                    oldgi,newgi,pold,pnew,nj1,nj2,xsumm1,
B                    xsumm2,fxgivkgl,fxgivkgn)
implicit none
integer KMAX,HYPMAX,XSMAX
integer nj(KMAX),oldgi,newgi,nj1,nj2,pold,pnew
double precision xsumm(XSMAX,KMAX),const(KMAX),xsumm1(XSMAX)
double precision xsumm2(XSMAX),phi(HYPMAX,KMAX)
double precision fxgivkgl,fxgivkgn

integer njj
double precision term

fxgivkgn=fxgivkgl

njj = nj(pold)
call comptermN(KMAX,HYPMAX,XSMAX,oldgi,pold,njj,phi,xsumm,const,
A              term)
fxgivkgn = fxgivkgn - term

njj = nj(pnew)
call comptermN(KMAX,HYPMAX,XSMAX,newgi,pnew,njj,phi,xsumm,const,
A              term)
fxgivkgn = fxgivkgn - term

call compterm1N(KMAX,HYPMAX,XSMAX,oldgi,nj1,const,phi,xsumm1,term)
fxgivkgn = fxgivkgn + term

call compterm1N(KMAX,HYPMAX,XSMAX,newgi,nj2,const,phi,xsumm2,term)
fxgivkgn = fxgivkgn + term

return
end



SUBROUTINE compterm1N(KMAX,HYPMAX,XSMAX,j,njj,const,phi,xsummj,
A                     term)
implicit none
integer KMAX,HYPMAX,XSMAX
integer j,njj
double precision const(KMAX),phi(HYPMAX,KMAX)
double precision xsummj(XSMAX),term

double precision gamnj2,taunj,gam2,quad
double precision xmean,x2mean

if(njj.gt.0) then
   xmean = xsummj(1)/dfloat(njj)
   x2mean = xsummj(2) - (xsummj(1) ** 2.0d-00) / dfloat(njj)
   if(x2mean.lt.0.0d-00) x2mean = 0.0d-00
   gamnj2=phi(3,j)+dfloat(njj)/2.0d-00
   taunj=phi(2,j)+dfloat(njj)
   call dlgama(gamnj2,gam2)
   quad=x2mean+((xmean-phi(1,j))**2)*phi(2,j)*njj/taunj
   term =   gam2
   A        + const(j)
   B        + 0.5d-00*dlog(phi(2,j)/taunj)
   C          - gamnj2*dlog(2.0d-00*phi(4,j)+quad)
else
   term=0.0d-00
endif

return
end



SUBROUTINE summetrN(NMAX,XSMAX,i,j,j1,nj1,nj2,x,xsumm1,xsumm2)
implicit none
integer NMAX,XSMAX,i,j,j1,nj1,nj2
double precision x(NMAX),xsumm1(XSMAX),xsumm2(XSMAX)

if(j.eq.j1) then
   nj1 = nj1 + 1
else
   nj2 = nj2 + 1
```

```
endif
if(j.eq.j1) then
   xsumm1(1) = xsumm1(1) + x(i)
   xsumm1(2) = xsumm1(2) + x(i)**2
else
   xsumm2(1) = xsumm2(1) + x(i)
   xsumm2(2) = xsumm2(2) + x(i)**2
endif

return
end



SUBROUTINE summejtN(KMAX,NMAX,XSMAX,dxsm,x,igchng,njted,
A                   ejectingp,xsumm,xsumting,xsumted)
implicit none
integer NMAX,KMAX,XSMAX
integer dxsm,njted,ejectingp,igchng(NMAX)
double precision x(NMAX),xsumm(XSMAX,KMAX)
double precision xsumting(XSMAX),xsumted(XSMAX)

integer i,m,h

do m=1,dxsm
   xsumting(m) = xsumm(m,ejectingp)
   xsumted(m) = 0.0d-00
enddo

do h=1,njted
   i = igchng(h)

   xsumting(1) = xsumting(1) - x(i)
   xsumting(2) = xsumting(2) - x(i)**2

   xsumted(1) = xsumted(1) + x(i)
   xsumted(2) = xsumted(2) + x(i)**2

enddo

return
end



SUBROUTINE fxgivkgejtN(KMAX,HYPMAX,XSMAX,nj,xsumm,const,phi,
A                      ejectingc,ejectedc,ejectingp,njting,
B                      njted,xsumting,xsumted,lfxgivkgrat)
implicit none
integer KMAX,HYPMAX,XSMAX
integer nj(KMAX),njting,njted
integer ejectingc,ejectedc,ejectingp
double precision xsumm(XSMAX,KMAX),const(KMAX),xsumting(XSMAX)
double precision xsumted(XSMAX),phi(HYPMAX,KMAX)
double precision lfxgivkgrat

integer j,njj
double precision term

j = ejectingc
njj = nj(ejectingp)
call comptermN(KMAX,HYPMAX,XSMAX,j,ejectingp,njj,phi,xsumm,
A              const,term)
lfxgivkgrat = -term

call compterm1N(KMAX,HYPMAX,XSMAX,j,njting,const,phi,xsumting,
A               term)
lfxgivkgrat = lfxgivkgrat + term

call compterm1N(KMAX,HYPMAX,XSMAX,ejectedc,njted,const,phi,
A               xsumted,term)
lfxgivkgrat = lfxgivkgrat + term

return
end



SUBROUTINE summabsN(KMAX,NMAX,XSMAX,absorbingp,absorbedp,xsumm,
A                   xsumabs)
implicit none
integer NMAX,KMAX,XSMAX
integer absorbingp,absorbedp
double precision xsumm(XSMAX,KMAX),xsumabs(XSMAX)

xsumabs(1) = xsumm(1,absorbingp) + xsumm(1,absorbedp)
xsumabs(2) = xsumm(2,absorbingp) + xsumm(2,absorbedp)

return
end



SUBROUTINE fxgivkgabsN(KMAX,HYPMAX,XSMAX,nj,xsumm,const,phi,
A                      absorbingc,absorbedc,absorbingp,absorbedp,
B                      njabs,xsumabs,lfxgivkgrat)
implicit none
integer KMAX,HYPMAX,XSMAX
integer nj(KMAX),njabs
integer absorbingc,absorbedc,absorbingp,absorbedp
double precision xsumm(XSMAX,KMAX),const(KMAX),xsumabs(XSMAX)
double precision phi(HYPMAX,KMAX)
double precision lfxgivkgrat

integer j,njj
double precision term

j = absorbedc
njj = nj(absorbedp)
call comptermN(KMAX,HYPMAX,XSMAX,j,absorbedp,njj,phi,xsumm,const,
A              term)
lfxgivkgrat = term

j = absorbingc
njj = nj(absorbingp)
call comptermN(KMAX,HYPMAX,XSMAX,j,absorbingp,njj,phi,xsumm,const,
A              term)
lfxgivkgrat = lfxgivkgrat + term

call compterm1N(KMAX,HYPMAX,XSMAX,j,njabs,const,phi,xsumabs,term)
lfxgivkgrat = lfxgivkgrat - term

return
end



SUBROUTINE selecxg2N(KMAX,NMAX,HYPMAX,XSMAX,x,njmove,
A                    gmove,jold,jnew,pold,pnew,nj,xsumm,idg,idgn,
B                    nj1,nj2,xsumm1,xsumm2)
implicit none
integer KMAX,NMAX,HYPMAX,XSMAX
integer njmove,gmove(NMAX),jold,jnew,pold,pnew
integer nj(KMAX),idg,idgn,nj1,nj2
double precision x(NMAX),xsumm(XSMAX,KMAX)
double precision xsumm1(XSMAX),xsumm2(XSMAX)

integer count,i

idgn=idg
if(nj(pold).eq.njmove) idgn=idgn-1
if(nj(pnew).eq.0) idgn=idgn+1
nj1 = nj(pold) - njmove
nj2 = nj(pnew) + njmove
do i=1,2
   xsumm1(i) = xsumm(i,pold)
   xsumm2(i) = xsumm(i,pnew)
enddo
do count=1,njmove
   i = gmove(count)
   xsumm1(1) = xsumm1(1) - x(i)
   xsumm1(2) = xsumm1(2) - x(i)**2
   xsumm2(1) = xsumm2(1) + x(i)
   xsumm2(2) = xsumm2(2) + x(i)**2
enddo

return
end



SUBROUTINE fxgivkgmlabN(KMAX,HYPMAX,XSMAX,j1,j2,p1,p2,nj1,nj2,
A                       phi,xsumm,const,fxgivkgl,fxgivkgn)
implicit none
integer KMAX,HYPMAX,XSMAX
integer j1,j2,p1,p2,nj1,nj2
```

```
double precision phi(HYPMAX,KMAX),xsumm(XSMAX,KMAX)
double precision const(KMAX),fxgivkgl,fxgivkgn

double precision term1,term2,term3,term4

call comptermN(KMAX,HYPMAX,XSMAX,j1,p1,nj1,phi,xsumm,const,term1)
call comptermN(KMAX,HYPMAX,XSMAX,j2,p2,nj2,phi,xsumm,const,term2)
call comptermN(KMAX,HYPMAX,XSMAX,j1,p2,nj2,phi,xsumm,const,term3)
call comptermN(KMAX,HYPMAX,XSMAX,j2,p1,nj1,phi,xsumm,const,term4)
fxgivkgn = fxgivkgl - term1 - term2 + term3 + term4

return
end



SUBROUTINE writehypN(KMAX,HYPMAX,phi,ouhyp)
implicit none
integer KMAX,HYPMAX
integer ouhyp
double precision phi(HYPMAX,KMAX)

write(ouhyp,*)  phi(2,1), phi(4,1)

return
end



SUBROUTINE assign(NMAX,KMAX,SAMPMAX,km,n,nsout,ouk,oug,ougr,
A                 filek,fileg,nfilnam)
implicit none
integer NMAX,KMAX,SAMPMAX
integer km,n,nsout,ouk,oug,ougr,nfilnam
character*24 filek,fileg

integer k(SAMPMAX),knempt(SAMPMAX)
integer g1(NMAX),g2(NMAX),costmat(130,131),totcost,perm(KMAX)
integer i,j,j1,j2,i1,i2,m1,m2,gi,iter,lowerbnd
integer table(KMAX),prev(KMAX),m
character gcharmat(SAMPMAX,NMAX)
character gchar(NMAX)
character*24 filegrel

open(ouk,file=filek,access='sequential',status='old')
open(oug,file=fileg,access='sequential',status='old')

do j=1,nsout
   read(ouk,1) k(j), knempt(j)
enddo
do j=1,nsout
   read(oug,2) (gcharmat(j,i),i=1,n)
enddo

close(ouk,status='keep')
close(oug,status='keep')

filegrel = fileg(1:nfilnam+4)//"rel"
open(ougr,file=filegrel,access='sequential',status='new')

do m1=1,km
   table(m1) = 0
   prev(m1) = 0
enddo
do j=1,nsout
   table(knempt(j)) = table(knempt(j)) + 1
enddo
do m=2,km
   if(table(m-1).ne.0) then
      prev(m) = m-1
   else
      prev(m) = prev(m-1)
   endif
enddo

do m1=1,km
   do iter=1,2
      do j1=1,nsout
         if(knempt(j1).eq.m1) then
            do i1=1,km
               do i2=1,km
                  costmat(i1,i2)=0
               enddo
            enddo
            do i=1,n
               gchar(i) = gcharmat(j1,i)
            enddo
            call gchar2int(NMAX,n,gchar,g1)
            lowerbnd = 0
            do j2=1,nsout
               if((knempt(j2).eq.prev(m1)).or.
               A  (knempt(j2).eq.m1.and.
               B  (iter.ge.2.or.j2.lt.j1) ) ) then
                  do i=1,n
                     gchar(i) = gcharmat(j2,i)
                  enddo
                  call gchar2int(NMAX,n,gchar,g2)
                  do i=1,n
                     do m2=1,knempt(j2)
                        if(m2.eq.g2(i)) then
                           costmat(m2,g1(i)) = costmat(m2,g1(i))-1
                           lowerbnd = lowerbnd + 1
                        endif
                     enddo
                  enddo
               endif
            enddo
            do i1=1,km
               do i2=1,km
                  costmat(i1,i2) = costmat(i1,i2) + lowerbnd
               enddo
            enddo
            call assct(m1,costmat,perm,totcost)
            do i=1,n
               call gchar2int(1,1,gcharmat(j1,i),gi)
               call gint2char(1,1,perm(gi),gchar)
               gcharmat(j1,i) = gchar(1)
            enddo
         endif
      enddo
   enddo
enddo

do j=1,nsout
   write(ougr,2) (gcharmat(j,i),i=1,n)
enddo

close(ougr,status='keep')

1    format(2i3)
2    format(1000000a1)

stop
end



SUBROUTINE gchar2int(NMAX,n,gchar,g)
implicit none
integer NMAX
integer n,g(NMAX)
character gchar(NMAX)

C Changes char g to an integer vector g (inverse of gint2char)

integer i,j
character*89 numb
data numb/"123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0abcdefghijklmnopqrs
1          tuvwxyz!%^&*()-_+={}[]~#@:;?><.,/|"/

do i=1,n
   do j=1,89
      if(numb(j:j).eq.gchar(i)) then
         g(i)=j
         go to 10
      endif
   enddo
10      continue
enddo

return
end



SUBROUTINE gint2char(NMAX,n,g,gchar)
```

```
implicit none
integer NMAX
integer n,g(NMAX)
character gchar(NMAX)

C Changes labels in g from 1:km to alphanumeric values

integer i,ig
character*89 numb
data numb/"123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0abcdefghijklmnopqrs
A          tuvwxyz!%^&*()-_+={}[]~#@:;?><.,/|"/

do i=1,n
   ig=g(i)
   gchar(i) = numb(ig:ig)
enddo

return
end



SUBROUTINE setseed(seed)
implicit none
integer seed(4)
integer i, idum(4)
common /seedstor/ idum

do i=1,4
   idum(i) = seed(i)
enddo

return
end
```

Bibliography

Aitkin, M. (2001). Likelihood and Bayesian analysis of mixtures. *Statistical Modelling, 1*, 287–304.

Andrieu, C., N. De Freitas, A. Doucet, and M. I. Jordan (2003). An Introduction to MCMC for Machine Learning. *Machine Learning, 50*, 5–43.

Basford, K. E., G. J. McLachlan, and M. G. York (1997). Modelling the distribution of stamp thickness via finite normal mixtures: The 1872 Hidalgo stamp issue of Mexico revisited. *Journal of Applied Statistics, 24*, 169–179.

Bechtel, Y. C., C. Bonaïti-Pellié, N. Poisson, J. Magnette, and P. R. Bechtel (1993). A population and family study of N-acetyltransferase using caffeine urinary metabolites. *Clinical Pharmacology and Therapeutics, 54*, 134–141.

Binder, D. A. (1978). Bayesian cluster analysis. *Biometrika, 65*, 31–38.

Cappé, O., C. P. Robert, and T. Rydén (1995). Bayesian model choice via Markov chain Monte Carlo methods. *Journal of Royal Statistical Society B, 57*, 473–484.

Carlin, B. P. and S. Chib (1995). Bayesian model choice via Markov chain Monte Carlo methods. *Journal of Royal Statistical Society B, 57*, 473–484.

Carpaneto, G. and P. Toth (1980). Algorithm 548: Solution of the assignment problem [H]. *ACM Transactions on Mathematical Software, 6*, 104–111.

Casella, G. and E. I. George (1992). Explaining the Gibbs Sampler. *The American Statistician, 46*, 167–174.

Casella, G., C. P. Robert, and M. T. Wells (2000). Mixture Models, Latent Variables and Partitioned Importance Sampling. Technical Report 2000-03, CREST, INSEE, Paris.

Celeux, G. (1998). *Bayesian inference for mixtures: the label switching problem* in COMPSTAT 98 - Procedures in Computational Statistics (eds R.Payne and P.J. Green). pp. 227-232, Heidelberg: Physica.

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

Chib, S. (1995). Marginal Likelihood from the Gibbs Output. *Journal of the American Statistical Association, 90*, 1313–1321.

Chib, S. and E. Greenberg (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician, 49*, 327–335.

Crawford, S. L., M. H. DeGroot, J. B. Kadane, and M. J. Small (1992). Modelling lake chemistry distributions: approximate Bayesian methods for estimating a finite-mixture model. *Technometrics, 34*, 441–453.

DeGroot, M. H. (1970). *Optimal Statistical Decisions*. USA: McGraw-Hill.

Dellaportas, P. and I. Papageorgiou (2006). Multivariate mixtures of normals with an unknown number of components. *Statistics and Computing, 16*, 57–68.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of Royal Statistical Society B, 39*, 1–38.

Diebolt, J. and C. P. Robert (1990). Bayesian estimation of finite mixture distributions: part II, Sampling implementation. Technical Report 111, Laboratoire de Statistique Théorique et Appliquée, Univerisité Paris VI, Paris.

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

Escobar, M. D. and M. West (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association, 90*, 577–588.

Everitt, B. S. and D. J. Hand (1981). *Finite Mixture Distributions*. London, UK: Chapman and Hall.

Fearnhead, P. (2004). Particle filters for mixture models with an unknown number of components. *Statistics and Computing, 14*, 11–21.

Ferguson, T. S. (1983). Bayesian Density Estimation by Mixtures of Normal Distributions. In *Recent Advances in Statistics* (eds H. Rizvi and J. Rustagi), New York, USA: Academic Press, 287–302.

Fisher, R. A. (1936). The Use of Multiple Measurements in Axonomic Problems. *Annals of Eugenics, 7*, 179–188.

Fryer, J. G. and C. A. Robertson (1972). A comparison of some methods for estimating mixed normal distributions. *Biometrika, 59*, 639–648.

Gelfand, A. E. and A. F. M. Smith (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association, 85*, 972–985.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 6*, 721–741.

Geyer, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science, 7*, 473–483.

Gilks, W. R., S. Richardson, and D. J. Spielgelhalter (1996). *Markov Chain Monte Carlo in Practice*. London, UK: Chapman & Hall.

Granger, C. W. J. and N. Newbold (1986). *Forecasting Economic Time Series*. London, UK: Academic Press Inc. (London) Ltd.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika, 82*, 711–732.

Green, P. J., N. L. Hjort, and S. Richardson (2003). *Highly Structured Stochastic Systems*. Oxford, UK: Oxford University Press.

Grenander, U. and M. I. Miller (1991). Jump-diffusion processes for abduction and recognition of biological shapes. Technical report, Electronic Signals and Systems Research Laboratory, Washington University.

Grenander, U. and M. I. Miller (1994). Representations of knowledge in complex systems. *Journal of Royal Statistical Society B, 56*, 549–603.

Gruet, M. A., A. Philippe, and C. P. Robert (1999). MCMC control spreadsheets for exponential mixture estimation. *Journal of Computational and Graphical Statistics, 8*, 298–317.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their application. *Biometrika, 57*, 97–109.

Hurn, M., A. Justel, and C. P. Robert (2003). Estimating Mixtures of Regressions. *Journal of Computational and Graphical Statistics, 12*, 55–79.

Ishwaran, H., L. F. James, and J. Sun (2001). Bayesian Model Selection in Finite Mixtures by Marginal Density Decompositions. *Journal of American Statistical Association, 96*, 1316–1332.

Izenman, A. J. and C. J. Sommer (1988). Philatelic Mixtures and Multimodal Densities. *Journal of American Statistical Association, 83*, 941–953.

Jasra, A., C. C. Holmes, and D. A. Stephens (2005). Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modelling. *Statistical Science, 20*, 50–67.

Levine, R. A. and G. Casella (2006). Optimizing Random Scan Gibbs Samplers. *Journal of Multivariate Analysis, 97*, 2071–2100.

Li, B. (2006). A new approach to cluster analysis: the clustering-function-based method. *Journal of the Royal Statistical Society B, 68*, 457–476.

Liu, C. and D. B. Rubin (1994). The ECME algorithm: a simple extension of EM and ECM with a faster montone convergence. *Biometrika, 81*, 633–648.

MacLachlan, G. and D. Peel (2000). *Finite Mixture Models*. New York, USA: John Wiley & Sons.

MacLachlan, G. J. (1997). Discussion of "On Bayesian analysis of mixtures with an unknown number of components" by S. Richardson and P.J. Green. *Journal of Royal Statistical Society B, 59*, 779–780.

MacLachlan, G. J. and K. E. Basford (1987). *Mixture Models*. New York, USA: Marcel Dekker, Inc.

MacLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and Extensions*. New York, USA: Wiley.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability,* Berkeley: University of California Press, USA, 281–297.

Marin, J., K. Mengersen, and C. P. Robert (2005). Bayesian Modelling and Inference on Mixtures of Distributions. In *Handbook of Statistics: Volume 25* (eds D. Dey and C.R. Rao), North-Holland.

Marron, J. S. and M. P. Wand (1992). Exact Mean Integrated Squared Error. *The Annals of Statistics, 20*, 712–736.

McGrory, C. A. (2005). *Variational Approximations in Bayesian Model Selection*. Ph. D. thesis, Department of Statistics, University of Glasgow, Glasgow.

Meligkotsidou, L. (2007). Bayesian multivariate Poisson mixtures with an unknown number of components. *Statistics and Computing, 17*, 93–107.

Meng, X. L. and D. B. Rubin (1993). Maximisation likelihood estimation via the ECM algorithm: A general framework. *Biometrika, 80*, 267–278.

Meng, X. L. and D. van Dyk (1997). The EM algorithm-an old folk song sung to a fast new tune (with discussion). *Journal of the Royal Statistical Society B, 59*, 511–567.

Mengersen, K. and C. P. Robert (1996). Testing for mixtures: A Bayesian Entropic Approach (with discussion). In *Bayesian Statistics 5* (eds J.M. Bernado, J.O. Berger, A.P. Dawid and A.F.M. Smith), Oxford: Oxford University Press, 225–276.

Metropolis, N., M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equations of state calculations by fast computing machines. *The Journal of Chemical Physics, 21*, 1087–1092.

Mueller, P. (1991). A generic approach to posterior integration and Bayesian sampling. Technical Report 91-09, Statistics Department, Purdue University.

Neal, R. M. and G. E. Hinton (1998). A view of the EM algorithm that justifies incremental, sparse and other variants. In *Learning in Graphical Models* (ed M.I. Jordan), Dordrecht: Kluwer, 355–368.

Nobile, A. (1994). *Bayesian Analysis of Finite Mixture Distributions*. Ph. D. thesis, Department of Statistics, Carnegie Mellon University, Pittsburgh.

Nobile, A. (2004). On the posterior distribution of the number of components in a finite mixture. *The Annals of Statistics, 32*, 2044–2073.

Nobile, A. (2005). Bayesian finite mixtures: a note on prior specification and posterior computation. Technical Report 05-03, Department of Statistics, University of Glasgow.

Nobile, A. and A. T. Fearnside (2007). Bayesian finite mixtures with an unknown number of components: the allocation sampler. *Statistics and Computing, 17*, 147–162.

Pearson, K. (1894). Contribution to the mathematical theory of evolution. *Philosophical Transactions of Royal Society A, 185*, 71–100.

Phillips, D. B. and A. F. M. Smith (1996). Bayesian model comparison via jump diffusions. In *Markov Chain Monte Carlo in Practice* (eds W.R. Gilks, S. Richardson and D.J. Spiegelhalter), Chapman & Hall, 215–239.

Postman, M., J. P. Huchra, and M. J. Geller (1986). Probes of large-scale structure in the Corona Borealis region. *The Astronomical Journal, 92*, 1238–1247.

Preston, C. J. (1976). Spatial birth-and-death processes. *Bulletin of the Institute of International Statistics, 46*, 371–391.

Pritchard, J. K., M. Stephens, and P. Donnelly (2000). Inference of Population Structure Using Multilocus Genotype Data. *Genetics, 155*, 945–959.

Raftery, A. E. (1996). Hypothesis testing and model selection. In *Markov Chain Monte Carlo in Practice* (eds W.R. Gilks, S. Richardson and D.J. Spiegelhalter), Chapman & Hall, 163–187.

Rao, C. R. (1948). The utilization of multiple measurements in problems of biological classification. *Journal of Royal Statistical Society B, 10*, 159–203.

Redner, R. A. and H. R. Walker (1984). Mixture densities, maximum likelihood and the EM algorithm. *Society for Industrial and Applied Mathemetics, 26*, 195–239.

Richardson, S. and P. J. Green (1997). On Bayesian analysis of mixtures with an unknown number of components. *Journal of Royal Statistical Society B, 59*, 731–792.

Robert, C. P., T. Rydén, and D. M. Titterington (2000). Bayesian Inference in Hidden Markov Models through the Reversible Jump Markov Chain Monte Carlo Method. *Journal of Royal Statistical Society B, 62*, 57–75.

Roeder, K. (1990). Density estimation with confidence sets exemplified by superclusters and voids in galaxies. *Journal of American Statistical Association, 85*, 617–624.

Roeder, K. and L. Wasserman (1997). Practical Bayesian density estimation using mixtures of normals. *Journal of American Statistical Association, 92*, 894–902.

Rydén, T., T. Teräsvirta, and S. Åsbrink (1998). Stylized facts of daily return series and the hidden markov model. *Journal of Applied Econometrics, 13*(3), 217–244.

Sahu, S. K. and R. C. H. Cheng (2003). A fast distance based approach for

determining the number of components in mixtures. *The Canadian Journal of Statistics, 31*, 3–22.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*, 461–464.

Spiegelhalter, D. J., N. G. Best, B. P. Carlin, and A. Van der Linde (2002). Bayesian measures of model complexity and fit (with discussion). *Journal of Royal Statistical Society B, 64*, 583–639.

Steele, R. J., A. E. Raftery, and M. J. Edmond (2003). Computing Normalizing Constants for Finite Mixture Models via Incremental Mixture Importance Sampling. Technical Report 436, Department of Statistics, University of Washington.

Stephens, M. (1997a). *Bayesian methods for mixtures of normal distributions.* Ph. D. thesis, Department of Statistics, University of Oxford, Oxford.

Stephens, M. (1997b). Discussion of "On Bayesian analysis of mixtures with an unknown number of components" by S. Richardson and P.J. Green. *Journal of Royal Statistical Society B, 59*, 768–769.

Stephens, M. (2000a). Bayesian analysis of mixture models with an unknown number of components - an alternative to reversible jump methods. *The Annals of Statistics, 28*, 40–74.

Stephens, M. (2000b). Dealing with label switching in mixture models. *Journal of Royal Statistical Society B, 62*, 795–809.

Tan, W. Y. and W. C. Chang (1972). Some comparisons of the method of moments and the method of maximum likelihood in estimating parameters of a mixture of two normal densities. *Journal of the American Statistical Association, 67*, 702–708.

Tanner, M. A. and W. H. Wong (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Society, 82*, 528–540.

Tierney, L. (1994). Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics, 22*, 1701–1728.

Titterington, D. M., A. F. M. Smith, and U. E. Makov (1985). *Statistical Analysis of Finite Mixture Distributions*. Chichester, UK: John Wiley & Sons.

Ward, J. H. (1963). Hierarchial Grouping to optimize an objective function. *Journal of American Statistical Association, 58*, 236–244.

Wolfe, J. H. (1971). A Monte Carlo study of the sampling distribution of the likelihood ratio for mixtures of multinormal distributions. *Technical Bulletin STB 72-2* (San Diego, US Naval Personnel and Training Research Laboratory).

Wruck, E., J. Achar, and J. Mazucheli (2001). Classification and discrimination for populations with mixture of multivariate normal distributions. *Revista de Matematica e Estatistica, 19*, 383–396.

Zhang, Z., K. L. Chan, and C. Chen (2004). Learning a multivariate Gaussian mixture model with the reversible jump MCMC algorithm. *Statistics and Computing, 14*, 343–355.