mixtools: An R Package for Analyzing Finite Mixture Models

Tatiana Benaglia
Pennsylvania State University

Didier Chauveau
Université d'Orléans

David R. Hunter
Pennsylvania State University

Derek S. Young
Pennsylvania State University

Abstract

The **mixtools** package for R provides a set of functions for analyzing a variety of finite mixture models. These functions include both traditional methods, such as EM algorithms for univariate and multivariate normal mixtures, and newer methods that reflect some recent research in finite mixture models. In the latter category, **mixtools** provides algorithms for estimating parameters in a wide range of different mixture-of-regression contexts, in multinomial mixtures such as those arising from discretizing continuous multivariate data, in nonparametric situations where the multivariate component densities are completely unspecified, and in semiparametric situations such as a univariate location mixture of symmetric but otherwise unspecified densities. Many of the algorithms of the **mixtools** package are EM algorithms or are based on EM-like ideas, so this article includes an overview of EM algorithms for finite mixture models.

Keywords: cutpoint, EM algorithm, mixture of regressions, model-based clustering, nonparametric mixture, semiparametric mixture, unsupervised clustering.

1. Introduction to finite mixtures and mixtools

Authors' note: The original version of this vignette was produced using an article that appears in the *Journal of Statistical Software* (URL: http://www.jstatsoft.org/); see Benaglia, Chauveau, Hunter, and Young (2009c).

Populations of individuals may often be divided into subgroups. Yet even when we observe characteristics of these individuals that provide information about their subgroup memberships, we may not actually observe these memberships *per se*. The basic goal of the tools in the **mixtools** package (version 0.4.3, as of this writing) for R (R Development Core Team 2009) is to examine a sample of measurements to discern and describe subgroups of individuals, even when there is no observable variable that readily indexes into which subgroup an individual properly belongs. This task is sometimes referred to as "unsupervised clustering" in the literature, and in fact mixture models may be generally thought of as comprising the subset of clustering methods known as "model-based clustering". The **mixtools** package is available from the Comprehensive R Archive Network at http://CRAN.R-project.org/package=mixtools.

Finite mixture models may also be used in situations beyond those for which clustering of

individuals is of interest. For one thing, finite mixture models give descriptions of entire subgroups, rather than assignments of individuals to those subgroups (though the latter may be accomplished using mixture models). Indeed, even the subgroups may not necessarily be of interest; sometimes finite mixture models merely provide a means for adequately describing a particular distribution, such as the distribution of residuals in a linear regression model where outliers are present.

Whatever the goal of the modeler when employing mixture models, much of the theory of these models involves the assumption that the subgroups are distributed according to a particular parametric form — and quite often this form is univariate or multivariate normal. While **mixtools** does provide tools for traditional fitting of finite mixtures of univariate and multivariate normal distributions, it goes well beyond this well-studied realm. Arising from recent research whose goal is to relax or modify the assumption of multivariate normality, **mixtools** provides computational techniques for finite mixture model analysis in which components are regressions, multinomial vectors arising from discretization of multivariate data, or even distributions that are almost completely unspecified. This is the main feature that distinguishes **mixtools** from other mixture-related R packages, also available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/`, such as **mclust** (Fraley and Raftery 2009) and **flexmix** (Leisch 2004; Grün and Leisch 2008). We briefly mention these two packages in Sections 2.3 and 5.3, respectively.

To make the mixture model framework more concrete, suppose the possibly vector-valued random variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are a simple random sample from a finite mixture of $m > 1$ arbitrary distributions, which we will call *components* throughout this article. The density of each $\mathbf{X}_i$ may be written

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \phi_j(\mathbf{x}_i), \quad \mathbf{x}_i \in \mathbb{R}^r, \tag{1}$$

where $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\phi}) = (\lambda_1, \ldots, \lambda_m, \phi_1, \ldots, \phi_m)$ denotes the parameter and the λ_m are positive and sum to unity. We assume that the ϕ_j are drawn from some family $\mathcal{F}$ of multivariate density functions absolutely continuous with respect to, say, Lebesgue measure. The representation (1) is not identifiable if no restrictions are placed on $\mathcal{F}$, where by "identifiable" we mean that $g_{\boldsymbol{\theta}}$ has a *unique* representation of the form (1) and we do not consider that "label-switching" — i.e., reordering the m pairs $(\lambda_1, \phi_1), \ldots, (\lambda_m, \phi_m)$ — produces a distinct representation.

In the next sections we will sometimes have to distinguish between *parametric* and more general *nonparametric* situations. This distinction is related to the structure of the family $\mathcal{F}$ of distributions to which the component densities ϕ_j in model (1) belong. We say that the mixture is *parametric* if $\mathcal{F}$ is a parametric family, $\mathcal{F} = \{\phi(\cdot|\boldsymbol{\xi}), \boldsymbol{\xi} \in \mathbb{R}^d\}$, indexed by a ($d$-dimensional) Euclidean parameter $\boldsymbol{\xi}$. A parametric family often used is the univariate Gaussian family $\mathcal{F} = \{\phi(\cdot|\mu, \sigma^2) = \text{density of } \mathcal{N}(\mu, \sigma^2), (\mu, \sigma^2) \in \mathbb{R} \times \mathbb{R}_*^+\}$, in which case the model parameter reduces to $\boldsymbol{\theta} = (\boldsymbol{\lambda}, (\mu_1, \sigma_1^2), \ldots, (\mu_m, \sigma_m^2))$. For the multivariate case, a possible parametric model is the *conditionally i.i.d. normal model*, for which $\mathcal{F} = \{\phi(\mathbf{x}_i) = \prod_{k=1}^r f(x_{ik}), f(t) \text{ density of } \mathcal{N}(\mu, \sigma^2)\}$ (this model is included in **mixtools**; see Section 6.1). An example of a (multivariate) nonparametric situation is $\mathcal{F} = \{\phi(\mathbf{x}_i) = \prod_{k=1}^r f(x_{ik}), f(t) \text{ a univariate density on } \mathbb{R}\}$, in which case $\boldsymbol{\theta}$ consists in a Euclidean part ($\boldsymbol{\lambda}$) and a nonparametric part ($f_1, \ldots, f_m$).

As a simple example of a dataset to which mixture models may be applied, consider the sample depicted in Figure 1. In the Old Faithful dataset, measurements give time in minutes

between eruptions of the Old Faithful geyser in Yellowstone National Park, USA. These data are included as part of the **datasets** package in R (R Development Core Team 2009); type `help("faithful")` in R for more details.

```
> library(mixtools)
> data(faithful)
> attach(faithful)

> hist(waiting, main="Time between Old Faithful eruptions",
+      xlab="Minutes",  ylab="", cex.main=1.5, cex.lab=1.5, cex.axis=1.4)
```

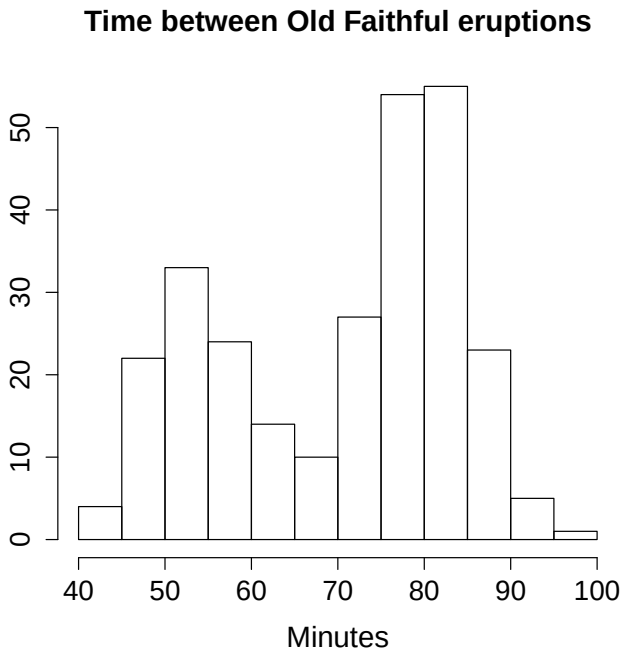

Figure 1: The Old Faithful dataset is clearly suggestive of a two-component mixture of symmetric components.

For the Old Faithful eruption data, a two-component mixture model is clearly a reasonable model based on the bimodality evident in the histogram. This example is analyzed by Hunter, Wang, and Hettmansperger (2007), who compare a standard normal-mixture method for fitting it with a novel semiparametric approach. Both approaches are included in **mixtools**; see Sections 2.3 and 4.2 of this article.

In Section 2 of the current article we review the well-known class of EM algorithms for finite mixture models, a common thread that runs throughout much of the rest of the article. The remaining sections discuss various categories of functions found in the **mixtools** package, from cutpoint methods that relax distributional assumptions for multivariate data by discretizing the data (Section 3), to semi- and non-parametric methods that eliminate distributional assumptions almost entirely depending on what the identifiability of the model allows (Section 4), to methods that handle various mixtures of regressions (Section 5). Finally, Section 6 describes several miscellaneous features of the **mixtools** package.

2. EM algorithms for finite mixtures

2.1. Missing data setup

Much of the general methodology used in **mixtools** involves the representation of the mixture problem as a particular case of maximum likelihood estimation (MLE) when the observations can be viewed as incomplete data. This setup implies consideration of two sample spaces, the sample space of the (incomplete) observations, and a sample space of some "complete" observations, the characterization of which being that the estimation can be performed explicitly at this level. For instance, in parametric situations, the MLE based on the complete data may exist in closed form. Among the numerous reference papers and monographs on this subject are, e.g., the original EM algorithm paper by Dempster, Laird, and Rubin (1977) and the finite mixture model book by McLachlan and Peel (2000) and references therein. We now give a brief description of this setup as it applies to finite mixture models in general.

The (observed) data consist of n i.i.d. observations $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ from a density $g_{\boldsymbol{\theta}}$ given by (1). It is common to denote the density of the sample by $\mathbf{g}_{\boldsymbol{\theta}}$, the n-fold product of $g_{\boldsymbol{\theta}}$, so that we write simply $\mathbf{x} \sim \mathbf{g}_{\boldsymbol{\theta}}$. In the missing data setup, $\mathbf{g}_{\boldsymbol{\theta}}$ is called the incomplete-data density, and the associated log-likelihood is $L_{\mathbf{x}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log g_{\boldsymbol{\theta}}(\mathbf{x}_i)$. The (parametric) ML estimation problem consists in finding $\hat{\boldsymbol{\theta}}_{\mathbf{x}} = \operatorname{argmax}_{\boldsymbol{\theta}\in\Phi} L_{\mathbf{x}}(\boldsymbol{\theta})$, or at least finding a local maximum — there are certain well-known cases in which a finite mixture model likelihood is unbounded (McLachlan and Peel 2000), but we ignore these technical details for now. Calculating $\hat{\boldsymbol{\theta}}_{\mathbf{x}}$ even for a parametric finite mixture model is known to be a difficult problem, and considering $\mathbf{x}$ as incomplete data resulting from non-observed complete data helps.

The associated complete data is denoted by $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$, with density $\mathbf{h}_{\boldsymbol{\theta}}(\mathbf{c}) = \prod_{i=1}^{n} h_{\boldsymbol{\theta}}(\mathbf{c}_i)$ (there exists a many-to-one mapping from $\mathbf{c}$ to $\mathbf{x}$, representing the loss of information). In the model for complete data associated with model (1), each random vector $\mathbf{C}_i = (\mathbf{X}_i, \mathbf{Z}_i)$, where $\mathbf{Z}_i = (Z_{ij}, j = 1, \ldots m)$, and $Z_{ij} \in \{0, 1\}$ is a Bernoulli random variable indicating that individual i comes from component j. Since each individual comes from exactly one component, this implies $\sum_{j=1}^{m} Z_{ij} = 1$, and

$$\mathsf{P}(Z_{ij} = 1) = \lambda_j, \quad (\mathbf{X}_i | Z_{ij} = 1) \sim \phi_j, \quad j = 1, \ldots, m.$$

The complete-data density for one observation is thus

$$h_{\boldsymbol{\theta}}(\mathbf{c}_i) = h_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{z}_i) = \sum_{j=1}^{m} \mathbb{I}_{z_{ij}} \lambda_j \phi_j(\mathbf{x}_i),$$

In the parametric situation, i.e. when $\mathcal{F}$ is a parametric family, it is easy to check that the complete-data MLE $\hat{\boldsymbol{\theta}}_{\mathbf{c}}$ based on maximizing $\log \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{c})$ is easy to find, provided that this is the case for the family $\mathcal{F}$.

2.2. EM algorithms

An EM algorithm iteratively maximizes, instead of the observed log-likelihood $L_{\mathbf{x}}(\boldsymbol{\theta})$, the operator

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathsf{E}\left[\log \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{C})|\mathbf{x}, \boldsymbol{\theta}^{(t)}\right],$$

where $\boldsymbol{\theta}^{(t)}$ is the current value at iteration t, and the expectation is with respect to the distribution $\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{c}|\mathbf{x})$ of $\mathbf{c}$ given $\mathbf{x}$, for the value $\boldsymbol{\theta}^{(t)}$ of the parameter. The iteration $\boldsymbol{\theta}^{(t)} \rightarrow \boldsymbol{\theta}^{(t+1)}$ is defined in the above general setup by

1. E-step: compute $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$
2. M-step: set $\boldsymbol{\theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}\in\Phi}\, Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$

For finite mixture models, the E-step does not depend on the structure of $\mathcal{F}$, since the missing data part is only related to the $\mathbf{z}$'s:

$$\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{c}|\mathbf{x}) = \prod_{i=1}^{n} k_{\boldsymbol{\theta}}(\mathbf{z}_i|\mathbf{x}_i).$$

The $\mathbf{z}$ are discrete, and their distribution is given via Bayes' theorem. The M-step itself can be split in two parts, the maximization related to $\boldsymbol{\lambda}$, which does not depend on $\mathcal{F}$, and the maximization related to $\boldsymbol{\phi}$, which has to be handled specifically (say, parametrically, semi- or non-parametrically) for each model. Hence the EM algorithms for the models handled by the **mixtools** package share the following common features:

1. **E-step:** Calculate the "posterior" probabilities (conditional on the data and $\boldsymbol{\theta}^{(t)}$) of component inclusion,

$$p_{ij}^{(t)} \stackrel{\text{def}}{=} \mathsf{P}_{\boldsymbol{\theta}^{(t)}}(Z_{ij} = 1|\mathbf{x}_i) = \frac{\lambda_j^{(t)}\phi_j^{(t)}(\mathbf{x}_i)}{\sum_{j'=1}^{m}\lambda_{j'}^{(t)}\phi_{j'}^{(t)}(\mathbf{x}_i)} \tag{2}$$

   for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Numerically, it can be dangerous to implement equation (2) exactly as written due to the possibility of the indeterminant form $0/0$ in cases where $\mathbf{x}_i$ is so far from any of the components that all $\phi_{j'}^{(t)}(\mathbf{x}_i)$ values result in a numerical underflow to zero. Thus, many of the routines in **mixtools** actually use the equivalent expression

$$p_{ij}^{(t)} = \left[1 + \sum_{j'\neq j}\frac{\lambda_{j'}^{(t)}\phi_{j'}^{(t)}(\mathbf{x}_i)}{\lambda_j^{(t)}\phi_j^{(t)}(\mathbf{x}_i)}\right]^{-1} \tag{3}$$

   or some variant thereof.

2. **M-step for $\boldsymbol{\lambda}$:** Set

$$\lambda_j^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} p_{ij}^{(t)}, \quad \text{for } j = 1, \ldots, m. \tag{4}$$

2.3. An EM algorithm example

As an example, we consider the univariate normal mixture analysis of the Old Faithful waiting data depicted in Figure 1. This fully parametric situation corresponds to a mixture from the univariate Gaussian family described in Section 1, where the jth component density $\phi_j(x)$ in

(1) is normal with mean μ_j and variance σ_j^2. This is a special case of the general mixture-of-normal model that is well-studied in the literature and for which other software, such as the **mclust** (Fraley and Raftery 2009) package for R, may also be used for parameter estimation. The M-step for the parameters (μ_j, σ_j^2), $j = 1, \ldots, m$ of this EM algorithm for such mixtures of univariate normals is straightforward, and can be found, e.g., in McLachlan and Peel (2000). The function `normalmixEM` implements the algorithm in **mixtools**. Code for the Old Faithful example, using most of the default values (e.g., stopping criterion, maximum number of iterations), is simply

```
> wait1 <- normalmixEM(waiting, lambda = .5, mu = c(55, 80), sigma = 5)

number of iterations= 9
```

The code above will fit a 2-component mixture (because `mu` is a vector of length two) in which the standard deviations are assumed equal (because `sigma` is a scalar instead of a vector). See `help("normalmixEM")` for details about specifying starting values for this EM algorithm.

```
> plot(wait1, density=TRUE, cex.axis=1.4, cex.lab=1.4, cex.main=1.8,
+      main2="Time between Old Faithful eruptions", xlab2="Minutes")
```

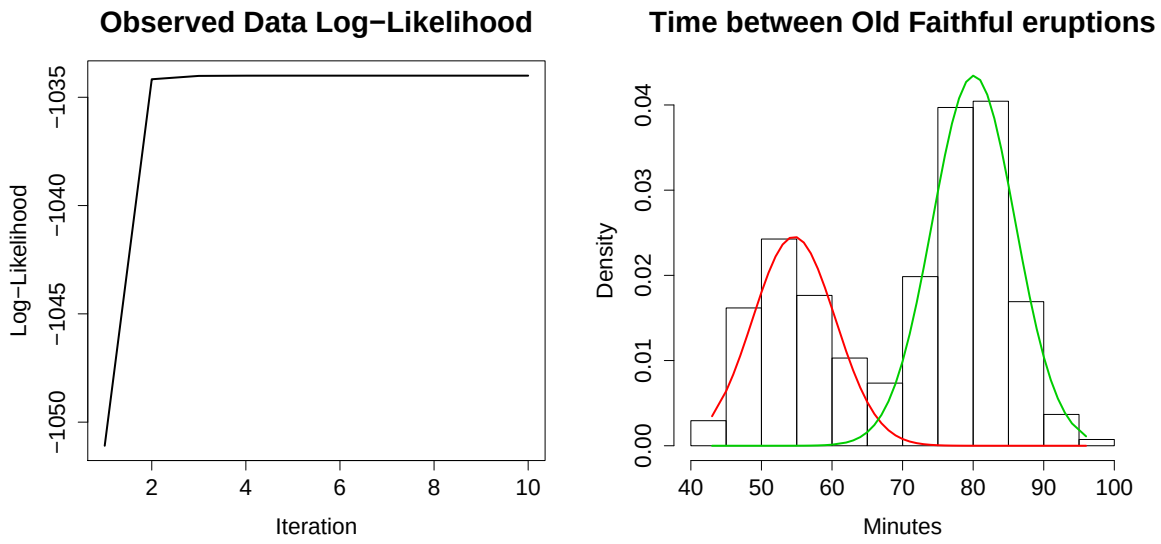

Figure 2: The Old Faithful waiting data fitted with a parametric EM algorithm in **mixtools**. Left: the sequence of log-likelihood values; Right: the fitted Gaussian components.

The `normalmixEM` function returns an object of class `"mixEM"`, and the `plot` method for these objects delivers the two plots given in Figure 2: the sequence $t \mapsto L_{\mathbf{x}}(\boldsymbol{\theta}^{(t)})$ of observed log-likelihood values and the histogram of the data with the m ($m = 2$ here) fitted Gaussian component densities of $\mathcal{N}(\hat{\mu}_j, \hat{\sigma}_j^2)$, $j = 1, \ldots, m$, each scaled by the corresponding $\hat{\lambda}_j$, superimposed. The estimator $\hat{\boldsymbol{\theta}}$ can be displayed by typing, e.g.,

```
> wait1[c("lambda", "mu", "sigma")]
```

```
$lambda
[1] 0.3608498 0.6391502

$mu
[1] 54.61364 80.09031

$sigma
[1] 5.869089 5.869089
```

Alternatively, the same output may be obtained using the `summary` method:

```
> summary(wait1)

summary of normalmixEM object:
         comp 1   comp 2
lambda  0.36085  0.63915
mu     54.61364 80.09031
sigma   5.86909  5.86909
loglik at estimate:  -1034.002
```

3. Cutpoint methods

Traditionally, most literature on finite mixture models has assumed that the density functions $\phi_j(\mathbf{x})$ of equation (1) come from a known parametric family. However, some authors have recently considered the problem in which $\phi_j(\mathbf{x})$ is unspecified except for some conditions necessary to ensure the identifiability of the parameters in the model. One such set of conditions is as follows:

Hettmansperger and Thomas (2000); Cruz-Medina, Hettmansperger, and Thomas (2004); and Elmore, Hettmansperger, and Thomas (2004) treat the case in which $\phi_j(\mathbf{x})$ equals the product $f_j(x_i) \cdots f_j(x_r)$ for some univariate density function f_j. Thus, conditional on knowing that $\mathbf{X}$ comes from the jth mixture component, the coordinates of $\mathbf{X}$ are independent and identically distributed. For this reason, this case is called the conditionally i.i.d. model.

The authors named above have developed an estimation method for the conditionally i.i.d. model. This method, the *cutpoint approach*, discretizes the continuous measurements by replacing each r-dimensional observation, say $\mathbf{X}_i = (x_{i1}, \ldots, x_{ir})$, by the p-dimensional multinomial vector $(n_1, \ldots, n_p)$, where $p \geq 2$ is chosen by the experimenter along with a set of cutpoints $-\infty = c_0 < c_1 < \cdots < c_p = \infty$, so that for $a = 1, \ldots, p$,

$$n_a = \sum_{k=1}^{r} I\{c_{a-1} < x_{ik} \leq c_a\}.$$

Note that the multinomial distribution is guaranteed by the conditional i.i.d. assumption, and the multinomial probability of the ath category is equal to $\theta_a \equiv P(c_{a-1} < X_{ik} \leq c_a)$.

The cutpoint approach is completely general in the sense that it can be applied to any number of components m and any number of repeated measures r, just as long as $r \geq 2m - 1$, a

condition that guarantees identifiability (Elmore and Wang 2003). However, some information is lost in the discretization step, and for this reason it becomes difficult to obtain density estimates of the component densities. Furthermore, even if the assumption of conditional independence is warranted, the extra assumption of identically distributed coordinates may not be; and the cutpoint method collapses when the coordinates are not identically distributed.

As an illustration of the cutpoint approach applied to a dataset, we show here how to use **mixtools** to reconstruct—almost—an example from Elmore *et al.* (2004). The dataset is `Waterdata`, a description of which is available by typing `help("Waterdata")`. This dataset contains 8 observations on each of 405 subjects, where the observations are angle degree measurements ranging from -90 to 90 that describe the subjects' answers to a series of 8 questions related to a conceptual task about how the surface of a liquid would be oriented if the vessel containing it were tipped to a particular angle. The correct answer is 0 degree in all cases, yet the subjects showed a remarkable variety of patterns of answers. Elmore *et al.* (2004) assumed the conditionally i.i.d. model (see Benaglia, Chauveau, and Hunter (2009a) for an in-depth discussion of this assumption and this dataset) with both $m = 3$ and $m = 4$ mixture components. Elmore *et al.* (2004) summarized their results by providing plots of estimated empirical distribution functions for the component distributions, where these functions are given by

$$\tilde{F}_j(x) = \frac{1}{mn\lambda_j} \sum_{i=1}^{n} \sum_{\ell=1}^{r} p_{ij} I\{x_{i\ell} \leq x\}. \tag{5}$$

In equation (5), the values of λ_j and p_{ij} are the final maximum likelihood estimates of the mixing proportions and posterior component membership probabilities that result from fitting a mixture of m multinomials (note in particular that the estimates of the multinomial parameters θ_a for each component are not used in this equation).

We cannot obtain the exact results of Elmore *et al.* (2004) because those authors do not state specifically which cutpoints c_a they use; they merely state that they use thirteen cutpoints. It appears from their Figures 1 and 2 that these cutpoints occur approximately at intervals of 10.5 degrees, starting at -63 and going through 63; these are the cutpoints that we adopt here. The function `makemultdata` will create a multinomial dataset from the original data, as follows:

```
> data("Waterdata")
> cutpts <- 10.5*(-6:6)
> watermult <- makemultdata(Waterdata, cuts = cutpts)
```

Once the multinomial data have been created, we may apply the `multmixEM` function to estimate the multinomial parameters via an EM algorithm.

```
> set.seed(15)
> theta4 <- matrix(runif(56), ncol = 14)
> theta3 <- theta4[1:3,]
> mult3 <- multmixEM(watermult, lambda = rep(1, 3)/3, theta = theta3)

number of iterations= 79

> mult4 <- multmixEM (watermult, lambda = rep (1, 4) / 4, theta = theta4)
```

```
number of iterations= 105
```

Finally, `compCDF` calculates and plots the estimated distribution functions of equation (5). Figure 3 gives plots for both a 3-component and a 4-component solution; these plots are very similar to the corresponding plots in Figures 1 and 2 of Elmore *et al.* (2004).

```
> cdf3 <- compCDF(Waterdata, mult3$posterior, lwd=2, lab=c(7, 5, 7),
+                 xlab="Angle in degrees", ylab="Component CDFs",
+                 main="Three-Component Solution")
> cdf4 <- compCDF(Waterdata, mult4$posterior, lwd=2, lab=c(7, 5, 7),
+                 xlab="Angle in degrees", ylab="Component CDFs",
+                 main="Four-Component Solution")
```

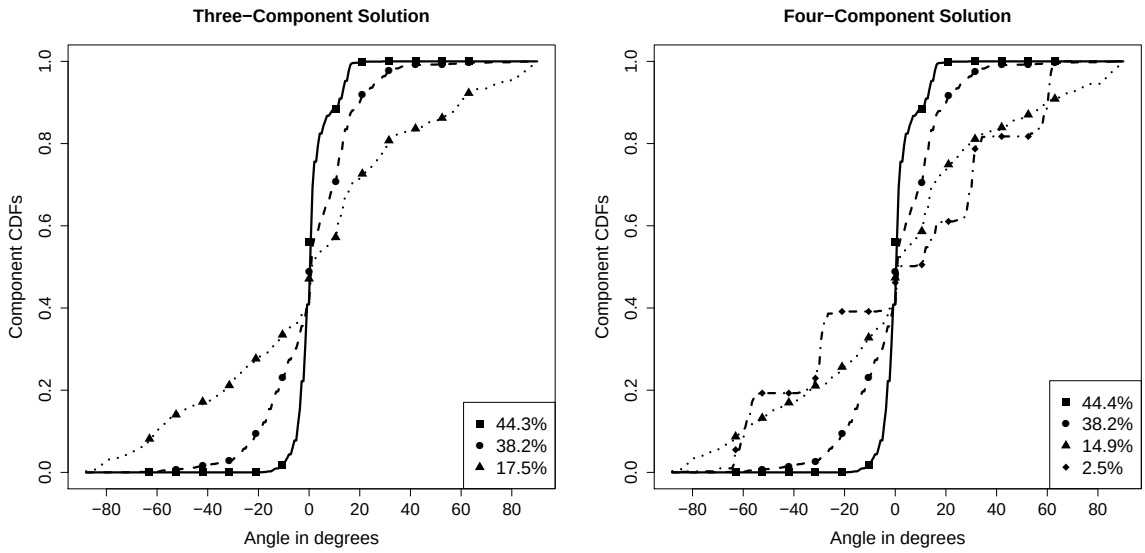

Figure 3: Empirical cumulative distribution function (CDF) estimates for the three- and four-component multinomial cutpoint models for the water-level data; compare Figures 1 and 2 of Elmore *et al.* (2004). The 13 cutpoints used are indicated by the points in the plots, and the estimated mixing proportions for the various components are given by the legend.

As with the output of `normalmixEM` in Section 2, it is possible to summarize the output of the `multmixEM` function using the `summary` method for `mixEM` objects:

```
> summary(mult4)

summary of multmixEM object:
              comp 1     comp 2    comp 3       comp 4
lambda   4.44218e-01 0.38163821 0.1486697  2.54743e-02
theta1  1.00000e-100 0.00124664 0.0870049  5.51164e-02
theta2  1.00000e-100 0.00549989 0.0453289  1.37695e-01
theta3  1.00000e-100 0.00870889 0.0374336  2.89421e-76
theta4  1.00000e-100 0.01201987 0.0409923  3.62476e-02
```

```
theta5   3.09495e-13 0.06709924 0.0457400  1.62337e-01
theta6   1.71175e-02 0.13605867 0.0712612 1.00000e-100
theta7   5.42656e-01 0.25723071 0.1459176  7.04952e-02
theta8   3.23932e-01 0.21853005 0.1128998  4.33644e-02
theta9   1.15712e-01 0.21095289 0.1626131  1.05242e-01
theta10  5.82600e-04 0.05923139 0.0617645  1.76528e-01
theta11 1.00000e-100 0.01470228 0.0284662  3.03943e-02
theta12  4.76749e-73 0.00087848 0.0309613  1.06723e-09
theta13  9.10041e-92 0.00454296 0.0384335  1.82580e-01
theta14  0.00000e+00 0.00329804 0.0911831  0.00000e+00
loglik at estimate:  -3094.588
```

4. Nonparametric and semiparametric methods

In this section, we consider nonparametric multivariate finite mixture models. The first algorithm presented here was introduced by Benaglia *et al.* (2009a) as a generalization of the stochastic semiparametric EM algorithm of Bordes, Chauveau, and Vandekerkhove (2007). Both algorithms are implemented in **mixtools**.

4.1. EM-like algorithms for mixtures of unspecified densities

Consider the mixture model described by equation (1). If we assume that the coordinates of the $\mathbf{X}_i$ vector are *conditionally independent*, i.e. they are independent conditional on the subpopulation or component (ϕ_1 through ϕ_m) from which $\mathbf{X}_i$ is drawn, the density in (1) can be rewritten as:

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \prod_{k=1}^{r} f_{jk}(x_{ik}), \tag{6}$$

where the function $f(\cdot)$, with or without subscripts, will always denote a univariate density function. Here we do not assume that $f_{jk}(\cdot)$ comes from a family of densities that may be indexed by a finite-dimensional parameter vector, and we estimate these densities using nonparametric density techniques. That is why we say that this algorithm is a fully nonparametric approach.

The density in equation (6) allows for a different distribution for each component and each coordinate of $\mathbf{X}_i$. Notice that if the density $f_{jk}(\cdot)$ does not depend on k, we have the case in which the $\mathbf{X}_i$ are not only conditionally independent but identically distributed as well. These are the two extreme cases. In order to encompass both the conditionally i.i.d. case and the more general case (6) simultaneously in one model, we allow that the coordinates of $\mathbf{X}_i$ are conditionally independent and there exist *blocks* of coordinates that are also identically distributed. If we let b_k denote the block to which the kth coordinate belongs, where $1 \leq b_k \leq B$ and B is the total number of such blocks, then equation (6) is replaced by

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \prod_{k=1}^{r} f_{jb_k}(x_{ik}). \tag{7}$$

The indices i, j, k, and ℓ will always denote a generic individual, component (subpopulation),

coordinate (repeated measurement), and block, respectively. Therefore, we will always have $1 \le i \le n$, $1 \le j \le m$, $1 \le k \le r$, and $1 \le \ell \le B$.

The EM algorithm to estimate model (7) has the E-step and M-step described in Section 2.2. In equation (2), we have $\phi_j^{(t)}(\mathbf{x}_i) = \prod_{k=1}^{r} f_{jb_k}^{(t)}(x_{ik})$, where $f_{j\ell}^{(t)}(\cdot)$ is obtained by a weighted nonparametric (kernel) density estimate, given by:

3. **Nonparametric (Kernel) density estimation step:** For any real u, define for each component $j \in \{1, \ldots, m\}$ and each block $\ell \in \{1, \ldots, B\}$

$$f_{j\ell}^{t+1}(u) = \frac{1}{n h_{j\ell} C_\ell \lambda_j^{t+1}} \sum_{k=1}^{r} \sum_{i=1}^{n} p_{ij}^{(t)} I\{b_k = \ell\} K\left(\frac{u - x_{ik}}{h_{j\ell}}\right), \tag{8}$$

where $K(\cdot)$ is a kernel density function, $h_{j\ell}$ is the bandwidth for the jth component and ℓth block density estimate, and C_ℓ is the number of coordinates in the ℓth block.

The function `npEM` implements this algorithm in **mixtools**. This function has an argument `samebw` which, when set to `TRUE` (the default), takes $h_{j\ell} = h$, for all $1 \le j \le m$ and $1 \le \ell \le B$, that is, the same bandwidth for all components and blocks, while `samebw = FALSE` allows a different bandwidth for each component and each block, as detailed in Benaglia, Chauveau, and Hunter (2009b). This function will, if called using `stochastic = TRUE`, replace the deterministic density estimation step (8) by a *stochastic* density estimation step of the type proposed by Bordes *et al.* (2007): First, generate $\mathbf{Z}_i^{(t)} = (Z_{i1}^{(t)}, \ldots, Z_{im}^{(t)})$ as a multivariate random vector with a single trial and success probability vector $\mathbf{p}_i^{(t)} = (p_{i1}^{(t)}, \ldots, p_{1m}^{(t)})$, then in the M-step for λ_j^{t+1} in equation (4), replace $p_{ij}^{(t)}$ by $Z_{ij}^{(t)}$ and let

$$f_{j\ell}^{t+1}(u) = \frac{1}{n h_{j\ell} C_\ell \lambda_j^{t+1}} \sum_{k=1}^{r} \sum_{i=1}^{n} Z_{ij}^{(t)} I\{b_k = \ell\} K\left(\frac{u - x_{ik}}{h_{j\ell}}\right).$$

In other words, the stochastic versions of these algorithms re-assign each observation randomly at each iteration, according to the $p_{ij}^{(t)}$ values at that iteration, to one of the m components, then the density estimate for each component is based only on those observations that have been assigned to it. Because the stochastic algorithms do not converge the way a deterministic algorithm often does, the output of `npEM` is slightly different when `stochastic = TRUE` than when `stochastic = FALSE`, the default. See the corresponding help file for details.

Benaglia *et al.* (2009a) also discuss specific cases of model (7) in which some of the $f_{jb_k}(\cdot)$ densities are assumed to be the same except for a location and scale change. They refer to such cases as semiparametric since estimating each $f_{jb_k}(\cdot)$ involves estimating an unknown density as well as multiple location and scale parameters. For instance, equation (17) of Benaglia *et al.* (2009a) sets

$$f_{j\ell}(x) = \frac{1}{\sigma_{j\ell}} f\left(\frac{x - \mu_{j\ell}}{\sigma_{j\ell}}\right), \tag{9}$$

where $\ell = b_k$ for a generic k.

The **mixtools** package implements an algorithm for fitting model (9) in a function called `spEM`. Details on the use of this function may be obtained by typing `help("spEM")`. Implementation of this algorithm and of that of the `npEM` function requires updating the values of $f_{jb_k}(x_{ik})$

for all i, j, and k for use in the E-step (2). To do this, the `spEM` algorithm keeps track of an $n \times m$ matrix, called Φ here, where

$$\Phi_{ij} \equiv \phi_j(\mathbf{x}_i) = \prod_{k=1}^{r} f_{jb_k}(x_{ik}).$$

The density estimation step of equation (8) updates the Φ matrix for the $(t+1)$th iteration based on the most recent values of all of the parameters. For instance, in the case of model (9), we obtain

$$\begin{aligned}
\Phi_{ij}^{t+1} &= \prod_{\ell=1}^{B} \prod_{k:b_k=\ell} \frac{1}{\sigma_{j\ell}^{t+1}} f^{t+1}\left(\frac{x-\mu_{j\ell}^{t+1}}{\sigma_{j\ell}^{t+1}}\right) \\
&= \prod_{\ell=1}^{B} \prod_{k:b_k=\ell} \frac{1}{\sigma_{j\ell}^{t+1}} \sum_{i'=1}^{n} \frac{p_{ij}^{t+1}}{hrn\lambda_j^{t+1}} \sum_{k'=1}^{r} K\left[\frac{\left(\frac{x_{ik}-\mu_{j\ell}^{t+1}}{\sigma_{j\ell}^{t+1}}\right) - (x_{i'k'} - \mu_{j\ell}^{t+1})}{h\sigma_{j\ell}^{t+1}}\right].
\end{aligned}$$

4.2. A univariate symmetric, location-shifted semiparametric example

Both Hunter *et al.* (2007) and Bordes, Mottelet, and Vandekerkhove (2006) study a particular case of model (1) in which x is univariate and

$$g_{\boldsymbol{\theta}}(x) = \sum_{j=1}^{m} \lambda_j \phi(x - \mu_j), \tag{10}$$

where $\phi(\cdot)$ is a density that is assumed to be completely unspecified except that it is symmetric about zero. Because each component distribution has both a nonparametric part $\phi(\cdot)$ and a parametric part μ_j, we refer to this model as semiparametric.

Under the additional assumption that $\phi(\cdot)$ is absolutely continuous with respect to Lebesgue measure, Bordes *et al.* (2007) propose a stochastic algorithm for estimating the model parameters, namely, $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \phi)$. This algorithm is implemented by the **mixtools** function `spEMsymloc`. This function also implements a nonstochastic version of the algorithm, which is the default and which is a special case of the general algorithm described in Section 4.1.

As noted in Figure 1, model (10) appears to be an appropriate model for the Old Faithful waiting times dataset. Here, we provide code that applies the `spEMsymloc` function to these data. First, we display the normal mixture solution of Figure 2 with a semiparametric solution superimposed, in Figure 4(a):

```
> plot(wait1, which = 2, cex.axis = 1.4, cex.lab = 1.4, cex.main = 1.8,
+      main2 = "Time between Old Faithful eruptions", xlab2 = "Minutes")
> wait2 <- spEMsymloc(waiting, mu0 = c(55, 80))
> plot(wait2, lty = 2, newplot = FALSE, addlegend = FALSE)
```

Because the semiparametric version relies on a kernel density estimation step (8), it is necessary to select a bandwidth for this step. By default, `spEMsymloc` uses a fairly simplistic approach: It applies "Silverman's rule of thumb" (Silverman 1986) to the entire dataset using the `bw.nrd0` function in R. For the Old Faithful waiting time dataset, this bandwidth is about 4:

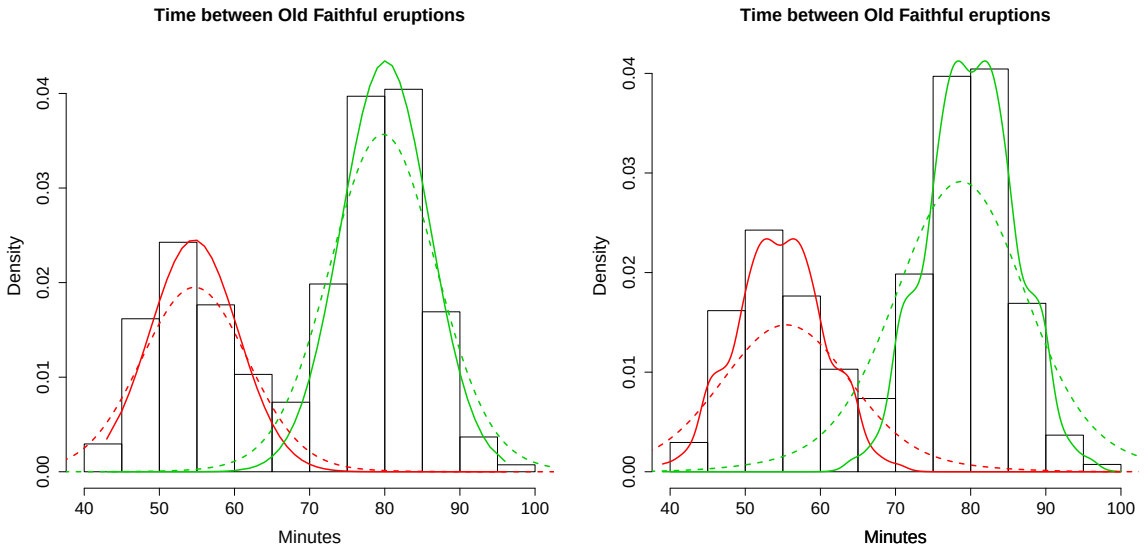

Figure 4: The Old Faithful dataset, fit using different algorithms in **mixtools**. Left: the fitted Gaussian components (solid) and a semiparametric fit assuming model (10) with the default bandwidth of 4.0 (dashed); Right: the same model (10) using bandwidths of 1.0 (solid) and 6.0 (dashed).

```
> bw.nrd0(waiting)

[1] 3.987559
```

But the choice of bandwidth can make a big difference, as seen in Figure 4(b).

```
> wait2a <- spEMsymloc(waiting, mu0 = c(55, 80), bw = 1)
> wait2b <- spEMsymloc(waiting, mu0 = c(55, 80), bw = 6)
> plot(wait2a, lty = 1, addlegend = FALSE, cex.axis = 1.4,
+      cex.lab = 1.4, cex.main = 1.8, xlab = "Minutes",
+      title = "Time between Old Faithful eruptions")
> plot(wait2b, lty = 2, newplot = FALSE, addlegend = FALSE)
```

We find that with a bandwidth near 2, the semiparametric solution looks quite close to the normal mixture solution of Figure 2. Reducing the bandwidth further results in the "bumpiness" exhibited by the solid line in Figure 4(b). On the other hand, with a bandwidth of 8, the semiparametric solution completely breaks down in the sense that algorithm tries to make each component look similar to the whole mixture distribution. We encourage the reader to experiment by changing the bandwidth in the above code.

4.3. A trivariate Gaussian example

As a first simple, nonparametric example, we simulate a Gaussian trivariate mixture with independent repeated measures and a shift of location between the two components in each coordinate, i.e., $m = 2$, $r = 3$, and $b_k = k$, $k = 1, 2, 3$. The individual densities f_{jk} are the

densities of $\mathcal{N}(\mu_{jk}, 1)$, with component means $\boldsymbol{\mu}_1 = (0, 0, 0)$ and $\boldsymbol{\mu}_2 = (3, 4, 5)$. This example was introduced by Hall, Neeman, Pakyari, and Elmore (2005) then later reused by Benaglia *et al.* (2009a) for comparison purposes. Note that the parameters in this model are identifiable, since Hall and Zhou (2003) showed that for two components ($m = 2$), identifiability holds in model (1) is under mild assumptions as long as $r \geq 3$, even in the most general case in which $b_k = k$ for all k.

A function `ise.npEM` has been included in **mixtools** for numerically computing the integrated squared error (ISE) relative to a user-specified true density for a selected estimated density $\hat{f}_{jk}$ from `npEM` output. Each density $\hat{f}_{jk}$ is computed using equation (8) together with the posterior probabilities after convergence of the algorithm, i.e., the final values of the p_{ij}^t (when `stochastic = FALSE`). We illustrate the usage of `ise.npEM` in this example by running a Monte Carlo simulation for S replications, then computing the square root of the mean integrated squared error (MISE) for each density, where

$$\text{MISE} = \frac{1}{S} \sum_{s=1}^{S} \int \left(\hat{f}_{jk}^{(s)}(u) - f_{jk}(u) \right)^2 du, \quad j = 1, 2 \text{ and } k = 1, 2, 3.$$

For this example, we first set up the model true parameters with $S = 100$ replications of $n = 300$ observations each:

```
> m <- 2; r <- 3; n <- 300; S <- 100
> lambda <- c(0.4, 0.6)
> mu <- matrix(c(0, 0, 0, 3, 4, 5), m, r, byrow = TRUE)
> sigma <- matrix(rep(1, 6), m, r, byrow = TRUE)
```

Next, we set up "arbitrary" initial centers, a matrix for storing sums of integrated squared errors, and an integer storing the number of suspected instances of label switching that may occur during the replications:

```
> centers <- matrix(c(0, 0, 0, 4, 4, 4), 2, 3, byrow = TRUE)
> ISE <- matrix(0, m, r, dimnames = list(Components = 1:m, Blocks = 1:r))
> nblabsw <- 0
```

Finally, we run the Monte Carlo simulation, using the `samebw = FALSE` option since it is more appropriate for this location-shift model:

```
> set.seed(1000)
> for (mc in 1:S) {
+   x <- rmvnormmix(n, lambda, mu, sigma)
+   a <- npEM(x, centers, verb = FALSE, samebw = FALSE)
+   if (a$lambda[1] > a$lambda[2]) nblabsw <- nblabsw + 1
+   for (j in 1:m) {
+     for (k in 1:r) {
+     ISE[j, k] <- ISE[j, k] + ise.npEM(a, j, k, dnorm,
+         lower = mu[j, k] - 5, upper = mu[j, k] + 5, plots = FALSE,
+         mean = mu[j, k], sd = sigma[j, k])$value
+     }
```

```
+   }
+ }
> MISE <- ISE/S
> print(sqMISE <- sqrt(MISE))

          Blocks
Components         1         2         3
         1 0.07724759 0.07699926 0.07745280
         2 0.06476345 0.06154020 0.06604921
```

We can examine the `npEM` output from the last replication above using

```
> summary(a)

300 observations, 3 coordinates, 2 components, and 3 blocks.

Means (and std. deviations) for each component:
  Block #1:  Coordinate 1
    -0.181 (0.892)   3.03 (0.961)
  Block #2:  Coordinate 2
    0.107 (1)   3.96 (1.04)
  Block #3:  Coordinate 3
    -0.086 (1.1)   5.09 (0.988)
```

We can also get plots of the estimated component densities for each block (recall that in this example, block ℓ consists only of coordinate ℓ) using the `plot` function. The resulting plots are given in Figure 5.

```
> plot(a)
```

4.4. A more general multivariate nonparametric example

In this section, we fit a more difficult example, with non-multimodal mixture densities (in block #2), heavy-tailed distributions, and different scales among the coordinates. The model is multivariate with $r = 5$ repeated measures and $m = 2$ components (hence identifiability holds; cf. Hall and Zhou (2003) as cited in Section 4.3). The 5 repeated measures are grouped into $B = 2$ blocks, with $b_1 = b_2 = b_3 = 1$ and $b_4 = b_5 = 2$. Block 1 corresponds to a mixture of two noncentral Student t distributions, $t'(2, 0)$ and $t'(10, 8)$, where the first parameter is the number of degrees of freedom, and the second is the non-centrality. Block 2 corresponds to a mixture of Beta distributions, $\mathcal{B}(1, 1)$ (which is actually the uniform distribution over $[0, 1]$) and $\mathcal{B}(1, 5)$. The first component weight is $\lambda_1 = 0.4$. The true mixtures are depicted in Figure 6.

To fit this model in **mixtools**, we first set up the model parameters:

```
> m <- 2; r <- 5
> lambda <- c(0.4, 0.6)
> df <- c(2, 10); ncp <- c(0, 8)
> sh1 <- c(1, 1) ; sh2 <- c(1, 5)
```

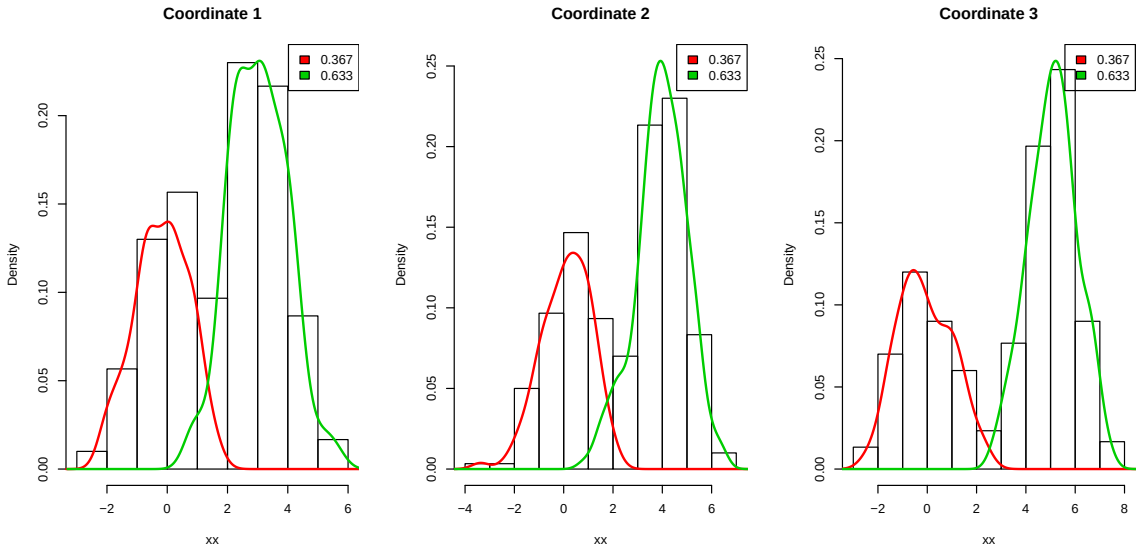

Figure 5: Output of the `npEM` algorithm for the trivariate Gaussian model with independent repeated measures.

Then we generate a pseudo-random sample of size $n = 300$ from this model:

```
> n <- 300; z <- sample(m, n, rep = TRUE, prob = lambda)
> r1 <- 3; z2 <- rep(z, r1)
> x1 <- matrix(rt(n * r1, df[z2], ncp[z2]), n, r1)
> r2 <- 2; z2 <- rep(z, r2)
> x2 <- matrix(rbeta(n * r2, sh1[z2], sh2[z2]), n, r2)
> x <- cbind(x1, x2)
```

For this example in which the coordinate densities are on different scales, it is obvious that the bandwidth in `npEM` should depend on the blocks and components. We set up the block structure and some initial centers, then run the algorithm with the option `samebw = FALSE`:

```
> id <- c(rep(1, r1), rep(2, r2))
> centers <- matrix(c(0, 0, 0, 1/2, 1/2, 4, 4, 4, 1/2, 1/2), m, r,
+                   byrow = TRUE)
> b <- npEM(x, centers, id, eps = 1e-8, verb = FALSE, samebw = FALSE)
```

Figure 7 shows the resulting density estimates, which may be obtained using the plotting function included in **mixtools**:

```
> plot(b, breaks = 15)
```

Finally, we can compute the ISE of the estimated density relative to the truth for each block and component. The corresponding output is depicted in Figure 8.

```
> par(mfrow=c(2,2))
> for (j in 1:2){
```

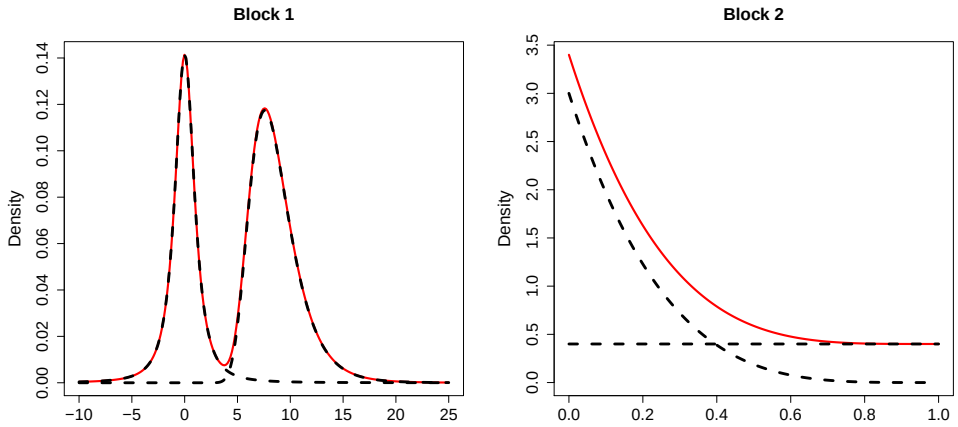

Figure 6: True densities for the mixture of Section 4.4, with individual component densities (scaled by λ_j) in dotted lines and mixture densities in solid lines. The noncentral t mixture of coordinates 1 through 3 is on the left, the beta mixture of coordinates 4 and 5 on the right.

```
+   ise.npEM(b, j, 1, truepdf = dt, lower = ncp[j] - 10,
+            upper = ncp[j] + 10, df = df[j], ncp = ncp[j])
+   ise.npEM(b, j, 2, truepdf = dbeta, lower = -0.5,
+            upper = 1.5,  shape1 = sh1[j], shape2 = sh2[j])
+ }
```

5. Mixtures of regressions

5.1. Mixtures of linear regressions

Consider a mixture setting where we now assume $\mathbf{X}_i$ is a vector of covariates observed with a response Y_i. The goal of mixtures of regressions is to describe the conditional distribution of $Y_i|\mathbf{X}_i$. Mixtures of regressions have been extensively studied in the econometrics literature and were first introduced by Quandt (1972) as the *switching regimes* (or *switching regressions*) problem. A switching regimes system is often compared to *structural change* in a system (Quandt and Ramsey 1978). A structural change assumes the system depends deterministically on some observable variables, but switching regimes implies one is unaware of what causes the switch between regimes. In the case where it is assumed there are two heterogeneous classes, Quandt (1972) characterized the switching regimes problem "by assuming that nature chooses between regimes with probabilities λ and $1-\lambda$".

Suppose we have n independent univariate observations, $y_1, \ldots, y_n$, each with a corresponding vector of predictors, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, with $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p})^\top$ for $i = 1, \ldots, n$. We often set $x_{i,1} = 1$ to allow for an intercept term. Let $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and let $\underline{\mathbf{X}}$ be the $n \times p$ matrix consisting of the predictor vectors.

Suppose further that each observation $(y_i, \mathbf{x}_i)$ belongs to one of m classes. Conditional on membership in the jth component, the relationship between y_i and $\mathbf{x}_i$ is the normal regression

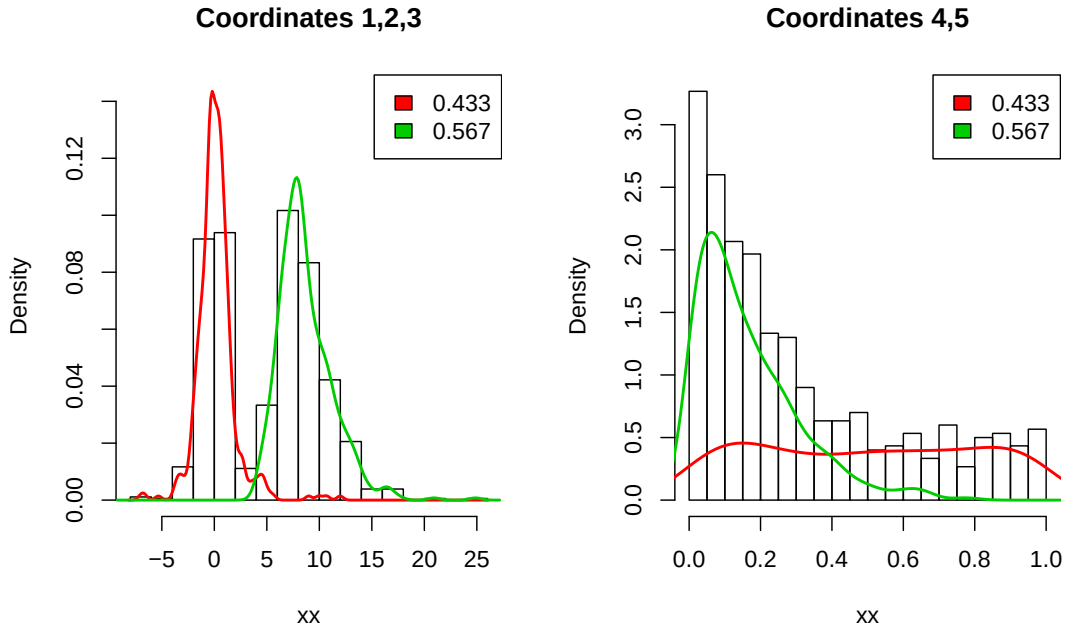

Figure 7: Result of plotting `npEM` output for the example of Section 4.4. Since $n = 300$, the histogram on the left includes 900 observations and the one on the right includes 600.

model

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta}_j + \epsilon_i, \tag{11}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_j^2)$ and $\boldsymbol{\beta}_j$ and σ_j^2 are the p-dimensional vector of regression coefficients and the error variance for component j, respectively.

Accounting for the mixture structure, the conditional density of $y_i|\mathbf{x}_i$ is

$$g_{\boldsymbol{\theta}}(y_i|\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \phi(y_i|\mathbf{x}_i^\top \boldsymbol{\beta}_j, \sigma_j^2), \tag{12}$$

where $\phi(\cdot|\mathbf{x}^\top \boldsymbol{\beta}_j, \sigma_j^2)$ is the normal density with mean $\mathbf{x}^\top \boldsymbol{\beta}$ and variance σ^2. Notice that the model parameter for this setting is $\boldsymbol{\theta} = (\boldsymbol{\lambda}, (\boldsymbol{\beta}_1, \sigma_1^2), \ldots, (\boldsymbol{\beta}_m, \sigma_m^2))$. The mixture of regressions model (12) differs from the well-known mixture of multivariate normals model $(Y_i, \mathbf{X}_i^\top)^\top \sim \sum_{j=1}^m \lambda_j \mathcal{N}_{p+1}(\boldsymbol{\mu}_j, \Sigma_j)$ because model (12) makes no assertion about the marginal distribution of $\mathbf{X}_i$, whereas the mixture of multivariate normals specifies that $\mathbf{X}_i$ itself has a mixture of multivariate normals distribution.

As a simple example of a dataset to which a mixture of regressions models may be applied, consider the sample depicted in Figure 9. In this dataset, the measurements of carbon dioxide (CO_2) emissions are plotted versus the gross national product (GNP) for $n = 28$ countries. These data are included **mixtools**; type `help("CO2data")` in R for more details. Hurn, Justel, and Robert (2003) analyzed these data using a mixture of regressions from the Bayesian perspective, pointing out that "there do seem to be several groups for which a linear model would be a reasonable approximation." They further point out that identification of such groups could clarify potential development paths of lower GNP countries.

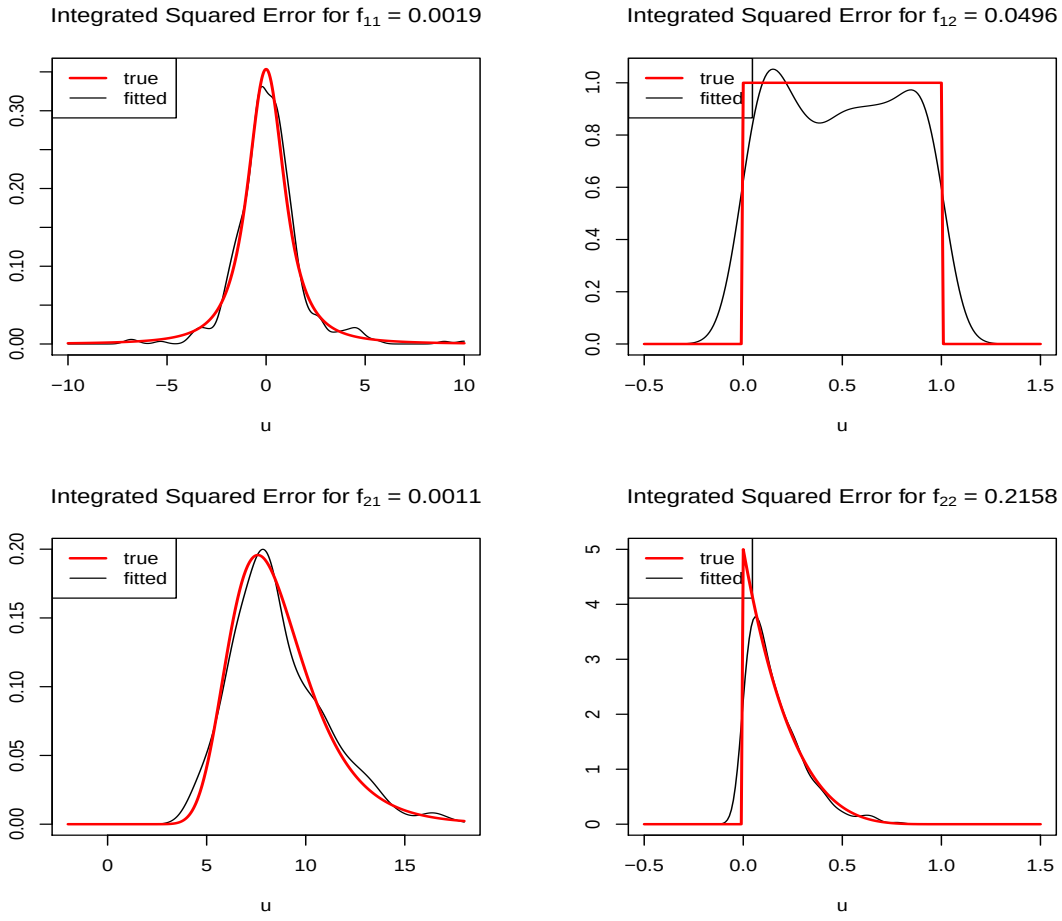

Figure 8: `ise.npEM` output for the 5-repeated measures example; the true densities are $f_{11} \equiv t'(2,0)$, $f_{21} \equiv t'(10,8)$, $f_{12} \equiv \mathcal{U}_{(0,1)}$, $f_{22} \equiv \mathcal{B}(1,5)$.

5.2. EM algorithms for mixtures of regressions

A standard EM algorithm, as described in Section 2, may be used to find a local maximum of the likelihood surface. The E-step is the same as for any finite mixture model EM algorithm; i.e., the $p_{ij}^{(t)}$ values are updated according to equation (2)—or, in reality, equation (3)—where each $\phi_j^{(t)}(\mathbf{x}_i)$ is replaced in the regression context by $\phi(y_i|\mathbf{x}_i^\top \boldsymbol{\beta}_j, \sigma_j^2)$:

$$p_{ij}^{(t)} = \left[1 + \sum_{j' \neq j} \frac{\lambda_{j'}^{(t)} \phi(y_i|\mathbf{x}_i^\top \boldsymbol{\beta}_{j'}, \sigma_{j'}^2)}{\lambda_j^{(t)} \phi(y_i|\mathbf{x}_i^\top \boldsymbol{\beta}_j, \sigma_j^2)}\right]^{-1} \tag{13}$$

The update to the λ parameters in the M-step, equation (4), is also the same. Letting $\mathbf{W}_j^{(t)} = \text{diag}(p_{1j}^{(t)}, \ldots, p_{nj}^{(t)})$, the additional M-step updates to the $\boldsymbol{\beta}$ and σ parameters are

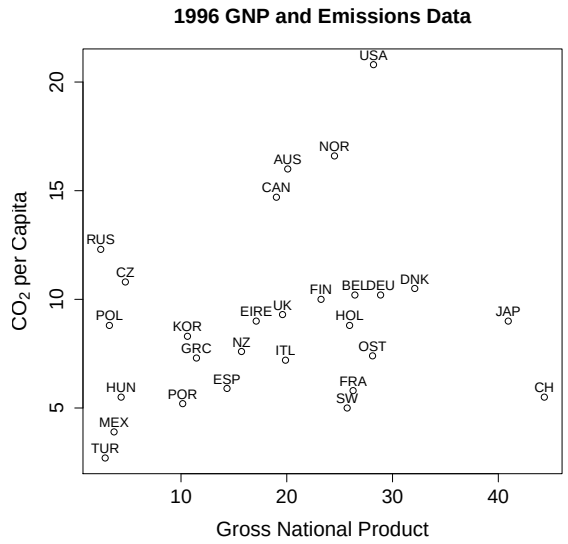

Figure 9: 1996 data on gross national product (GNP) per capita and estimated carbon dioxide (CO_2) emissions per capita. Note that "CH" stands for Switzerland, not China.

given by

$$\boldsymbol{\beta}_j^{(t+1)} = (\underline{\mathbf{X}}^\top \mathbf{W}_j^{(t)} \underline{\mathbf{X}})^{-1} \underline{\mathbf{X}}^\top \mathbf{W}_j^{(t)} \mathbf{y} \quad \text{and} \tag{14}$$

$$\sigma_j^{2(t+1)} = \frac{\left\| \mathbf{W}_j^{1/2(t)} (\mathbf{y} - \underline{\mathbf{X}}^\top \boldsymbol{\beta}_j^{(t+1)}) \right\|^2}{\mathrm{tr}(\mathbf{W}_j^{(t)})}, \tag{15}$$

where $\|\mathbf{A}\|^2 = \mathbf{A}^\top \mathbf{A}$ and $\mathrm{tr}(\mathbf{A})$ means the trace of the matrix $\mathbf{A}$. Notice that equation (14) is a weighted least squares (WLS) estimate of $\boldsymbol{\beta}_j$ and equation (15) resembles the variance estimate used in WLS.

Allowing each component to have its own error variance σ_j^2 results in the likelihood surface having no maximizer, since the likelihood may be driven to infinity if one component gives a regression surface passing through one or more points exactly and the variance for that component is allowed to go to zero. A similar phenomenon is well-known in the finite mixture-of-normals model where the component variances are allowed to be distinct (McLachlan and Peel 2000). However, in practice we observe this behavior infrequently, and the **mixtools** functions automatically force their EM algorithms to restart at randomly chosen parameter values when it occurs. A local maximum of the likelihood function, a consistent version of which is guaranteed to exist by the asymptotic theory as long as the model is correct and all λ_j are positive, usually results without any restarts.

The function `regmixEM` implements the EM algorithm for mixtures of regressions in **mixtools**. This function has arguments that control options such as adding an intercept term, `addintercept = TRUE`; forcing all $\boldsymbol{\beta}_j$ estimates to be the same, `arbmean = FALSE` (for instance, to model outlying observations as having a separate error variance from the non-outliers); and forcing all σ_j^2 estimates to be the same, `arbvar = FALSE`. For additional details, type `help("regmixEM")`.

As an example, we fit a 2-component model to the GNP data shown in Figure 9. Hurn *et al.* (2003) and Young (2007) selected 2 components for this dataset using model selection criteria, Bayesian approaches to selecting the number of components, and a bootstrapping approach. The function `regmixEM` will be used for fitting a 2-component mixture of regressions by an EM algorithm:

```
> data("CO2data")
> attach(CO2data)

> CO2reg <- regmixEM(CO2, GNP, lambda = c(1, 3) / 4,
+                    beta = matrix(c(8, -1, 1, 1), 2, 2), sigma = c(2, 1))

number of iterations= 10
```

We can then pull out the final observed log-likelihood as well as estimates for the 2-component fit, which include $\hat{\lambda}$, $\hat{\boldsymbol{\beta}}_1$, $\hat{\boldsymbol{\beta}}_2$, $\hat{\sigma}_1$, and $\hat{\sigma}_2$:

```
> summary(CO2reg)

summary of regmixEM object:
          comp 1   comp 2
lambda  0.754921 0.245079
sigma   2.049315 0.809389
beta1   8.678987 1.415150
beta2  -0.023344 0.676596
loglik at estimate:  -66.93977
```

The reader is encouraged to alter the starting values or let the internal algorithm generate random starting values. However, this fit seems appropriate and the solution is displayed in Figure 10 along with 99% Working-Hotelling Confidence Bands, which are constructed automatically by the `plot` method in this case by assigning each point to its most probable component and then fitting two separate linear regressions:

```
> plot(CO2reg, density = TRUE, alpha = 0.01, cex.main = 1.5, cex.lab = 1.5,
+      cex.axis = 1.4)
```

5.3. Predictor-dependent mixing proportions

Suppose that in model (12), we replace λ_j by $\lambda_j(\mathbf{x}_i)$ and assume that the mixing proportions vary as a function of the predictors $\mathbf{x}_i$. Allowing this type of flexibility in the model might be useful for a number of reasons. For instance, sometimes it is the proportions λ_j that are of primary scientific interest, and in a regression setting it may be helpful to know whether these proportions appear to vary with the predictors. As another example, consider a `regmixEM` model using `arbmean = FALSE` in which the mixture structure only concerns the error variance: In this case, $\lambda_j(\mathbf{x})$ would give some sense of the proportion of outliers in various regions of the predictor space.

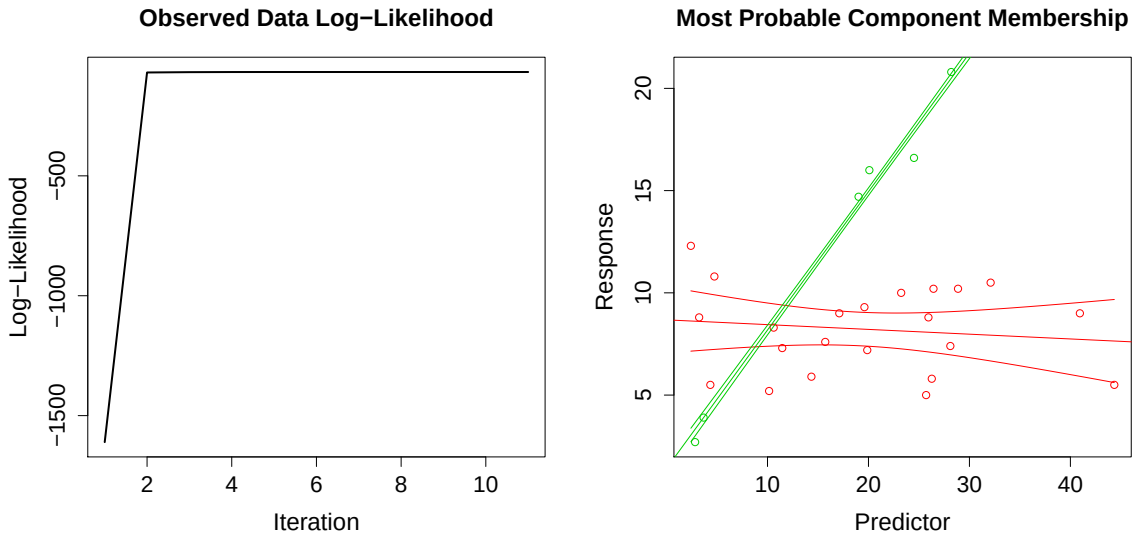

Figure 10: The GNP data fitted with a 2-component parametric EM algorithm in **mixtools**. Left: the sequence of log-likelihood values, $L_{\mathbf{x}}(\boldsymbol{\theta}^{(t)})$; Right: the fitted regression lines with 99% Working-Hotelling Confidence Bands.

One may assume that $\lambda_j(\mathbf{x})$ has a particular parametric form, such as a logistic function, which introduces new parameters requiring estimation. This is the idea of the *hierarchical mixtures of experts* (HME) procedure (Jacobs, Jordan, Nowlan, and Hinton 1991), which is commonly used in neural networks and which is implemented, for example, in the **flexmix** package for R (Leisch 2004; Grün and Leisch 2008). However, a parametric form of $\lambda_j(\mathbf{x})$ may be too restrictive; in particular, the logistic function is monotone, which may not realistically capture the pattern of change of λ_j as a function of $\mathbf{x}$. As an alternative, Young and Hunter (2009) propose a nonparametric estimate of $\lambda_j(\mathbf{x}_i)$ that uses ideas from kernel density estimation.

The intuition behind the approach of Young and Hunter (2009) is as follows: The M-step estimate (4) of λ_j at each iteration of a finite mixture model EM algorithm is simply an average of the "posterior" probabilities $p_{ij} = \mathsf{E}(Z_{ij}|\text{data})$. As a substitute, the nonparametric approach uses local linear regression to approximate the $\lambda_j(\mathbf{x})$ function. Considering the case of univariate x for simplicity, we set $\lambda_j(x) = \hat{\alpha}_{0j}(x)$, where

$$(\hat{\alpha}_{0j}(x), \hat{\alpha}_{1j}(x)) = \arg \min_{(\alpha_0,\alpha_1)} \sum_{i=1}^{n} K_h(x - x_i) \left[p_{ij} - \alpha_0 - \alpha_1(x - x_i)\right]^2 \tag{16}$$

and $K_h(\cdot)$ is a kernel density function with scale parameter (i.e., bandwidth) h. It is straightforward to generalize equation (16) to the case of vector-valued $\mathbf{x}$ by using a multivariate kernel function.

Young and Hunter (2009) give an iterative algorithm for estimating mixture of regression parameters that replaces the standard λ_j updates (4) by the kernel-weighted version (16). The algorithm is otherwise similar to a standard EM; thus, like the algorithm in Section 4.1 of this article, the resulting algorithm is an EM-like algorithm. Because only the λ_j parameters depend on $\mathbf{x}$ (and are thus "locally estimated"), whereas the other parameters (the $\boldsymbol{\beta}_j$ and

σ_j) can be considered to be globally estimated, Young and Hunter (2009) call this algorithm an iterative global/local estimation (IGLE) algorithm. Naturally, it replaces the usual E-step (13) by a modified version in which each λ_j is replaced by $\lambda_j(x_i)$.

The function `regmixEM.loc` implements the IGLE algorithm in **mixtools**. Like the `regmixEM` function, `regmixEM.loc` has the flexibility to include an intercept term by using `addintercept = TRUE`. Moreover, this function has the argument `kern.l` to specify the kernel used in the local estimation of the $\lambda_j(\mathbf{x}_i)$. Kernels the user may specify include `"Gaussian"`, `"Beta"`, `"Triangle"`, `"Cosinus"`, and `"Optcosinus"`. Further numeric arguments relating to the chosen kernel include `kernl.g` to specify the shape parameter for when `kern.l = "Beta"` and `kernl.h` to specify the bandwidth which controls the size of the window used in the local estimation of the mixing proportions. See the corresponding help file for additional details.

For the GNP and emissions dataset, Figure 10 indicates that the assumption of constant weights for the component regressions across all values of the covariate space may not be appropriate. The countries with higher GNP values appear to have a greater probability of belonging to the first component (i.e., the red line in Figure 10). We will therefore apply the IGLE algorithm to this dataset.

We will use the triweight kernel in equation (16), which is given by setting $\gamma = 3$ in

$$K_h(x) = \frac{1}{hB(1/2, \gamma + 1)} \left(1 - \frac{x^2}{h^2}\right)_+^{\gamma}, \tag{17}$$

where $B(x, y) = \Gamma(x)\Gamma(y)/\Gamma(x+y)$ is the beta function. For the triweight, $B(1/2, 4)$ is exactly $32/35$. This kernel may be specified in `regmixEM.loc` with `kern.l = "Beta"` and `kernl.g = 3`. The bandwidth we selected was $h = 20$, which we specify with `kernl.h = 20`.

For this implementation of the IGLE algorithm, we set the parameter estimates obtained from the mixture of regressions EM algorithm as starting values for $\hat{\boldsymbol{\beta}}_1$, $\hat{\boldsymbol{\beta}}_2$, $\hat{\sigma}_1$, and $\hat{\sigma}_2$, and set the starting values for $\lambda(x_i)$ to be 0.5 for all x_i.

```
> CO2igle <- regmixEM.loc(CO2, GNP, beta = CO2reg$beta, sigma = CO2reg$sigma,
+                         lambda = matrix(.5, 28, 2), kern.l = "Beta",
+                         kernl.h = 20, kernl.g = 3)

number of overall iterations= 14
```

We can view the estimates for $\hat{\boldsymbol{\beta}}_1$, $\hat{\boldsymbol{\beta}}_2$, $\hat{\sigma}_1$, and $\hat{\sigma}_2$. Notice that the estimates are comparable to those obtained for the mixture of regressions EM output and the log-likelihood value is slightly higher.

```
> summary(CO2igle)

summary of regmixEM.loc object:
          comp 1   comp 2
sigma  2.0168419 0.819108
beta1  8.8764980 1.504235
beta2 -0.0303274 0.672675
loglik at estimate:  -65.91835
```

Next, we can plot the estimates of $\lambda(x_i)$ from the IGLE algorithm.

```
> plot(GNP, CO2igle$post[,1], xlab = "GNP", cex.axis = 1.4, cex.lab = 1.5,
+   ylab = "Final posterior probabilities")
> lines(sort(GNP), CO2igle$lambda[order(GNP), 1], col=2)
> abline(h = CO2igle$lambda[1], lty = 2)
```

This plot is given in Figure 11. Notice the curvature provided by the estimates from the IGLE fit. These fits indicate an upward trend in the posteriors. The predictor-dependent mixing proportions model provides a viable way to reveal this trend since the regular mixture of regressions fit simply provides the same estimate of λ for all x_i.

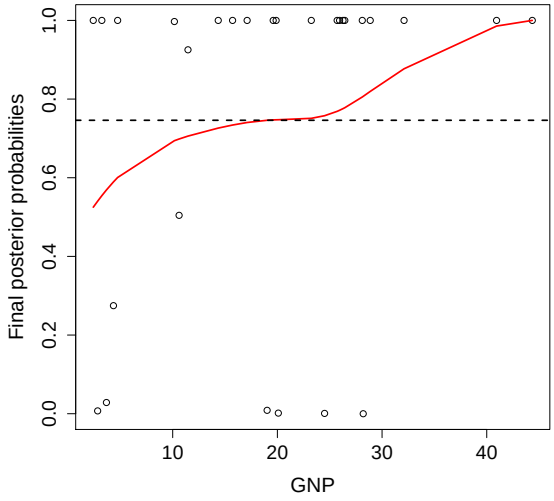

Figure 11: Posterior membership probabilities p_{i1} for component one versus the predictor GNP along with estimates of $\lambda_1(x)$ from the IGLE algorithm (the solid red curve) and λ_1 from the mixture of linear regressions EM algorithm (the dashed black line).

5.4. Parametric bootstrapping for standard errors

With likelihood methods for estimation in mixture models, it is possible to obtain standard error estimates by using the inverse of the observed information matrix when implementing a Newton-type method. However, this may be computationally burdensome. An alternative way to report standard errors in the likelihood setting is by implementing a parametric bootstrap. Efron and Tibshirani (1993) claim that the parametric bootstrap should provide similar standard error estimates to the traditional method involving the information matrix. In a mixture-of-regressions context, a parametric bootstrap scheme may be outlined as follows:

1. Use `regmixEM` to find a local maximizer $\hat{\boldsymbol{\theta}}$ of the likelihood.

2. For each $\mathbf{x}_i$, simulate a response value y_i^* from the mixture density $g_{\hat{\boldsymbol{\theta}}}(\cdot|\mathbf{x}_i)$.

3. Find a parameter estimate $\tilde{\boldsymbol{\theta}}$ for the bootstrap sample using `regmixEM`.

4. Use some type of check to determine whether label-switching appears to have occurred, and if so, correct it.

5. Repeat steps 2 through 4 B times to simulate the bootstrap sampling distribution of $\hat{\boldsymbol{\theta}}$.

6. Use the sample covariance matrix of the bootstrap sample as an approximation to the covariance matrix of $\hat{\boldsymbol{\theta}}$.

Note that step 3, which is not part of a standard parametric bootstrap, can be especially important in a mixture setting.

The **mixtools** package implements a parametric bootstrap algorithm in the `boot.se` function. We may apply it to the regression example of this section, which assumes the same estimate of λ for all x_i, as follows:

```
> set.seed(123)
> CO2boot <- boot.se(CO2reg, B = 100)
```

This output consists of both the standard error estimates and the parameter estimates obtained at each bootstrap replicate. An examination of the slope and intercept parameter estimates of the 500 bootstrap replicates reveals that no label-switching is likely to have occurred. For instance, the intercept terms of component one range from 4 to 11, whereas the intercept terms of component two are all tightly clumped around 0:

```
> rbind(range(CO2boot$beta[1,]), range(CO2boot$beta[2,]))
```

```
          [,1]        [,2]
[1,]  5.945735 11.37310064
[2,] -0.129666  0.08255873
```

We may examine the bootstrap standard error estimates by themselves as follows:

```
> CO2boot[c("lambda.se", "beta.se", "sigma.se")]
```

```
$lambda.se
[1] 0.08193493 0.08193493

$beta.se
           [,1]       [,2]
[1,] 1.04036016 1.19905882
[2,] 0.04145607 0.05397191

$sigma.se
[1] 0.3374287 0.3104505
```

6. Additional capabilities of mixtools

6.1. Selecting the number of components

Determining the number of components k is still a major contemporary issue in mixture modeling. Two commonly employed techniques are information criterion and parametric bootstrapping of the likelihood ratio test statistic values for testing

$$\begin{aligned} H_0 &: k = k_0 \\ H_1 &: k = k_0 + 1 \end{aligned} \tag{18}$$

for some positive integer k_0 (McLachlan 1987).

The **mixtools** package has functions to employ each of these methods using EM output from various mixture models. The information criterion functions calculate An Information Criterion (AIC) of Akaike (1973), the Bayesian Information Criterion (BIC) of Schwarz (1978), the Integrated Completed Likelihood (ICL) of Biernacki, Celeux, and Govaert (2000), and the consistent AIC (CAIC) of Bozdogan (1987). The functions for performing parametric bootstrapping of the likelihood ratio test statistics sequentially test $k = k_0$ versus $k = k_0 + 1$ for $k_0 = 1, 2, \ldots$, terminating after the bootstrapped p-value for one of these tests exceeds a specified significance level.

Currently, **mixtools** has functions for calculating information criteria for mixtures of multinomials (`multmixmodel.sel`), mixtures of multivariate normals under the conditionally i.i.d. assumption (`repnormmixmodel.sel`), and mixtures of regressions (`regmixmodel.sel`). Output from various mixture model fits available in **mixtools** can also be passed to the function `boot.comp` for the parametric bootstrapping approach. The parameter estimates from these EM fits are used to simulate data from the null distribution for the test given in (18). For example, the following application of the `multmixmodel.sel` function to the water-level multinomial data from Section 3 indicates that either 3 or 4 components seems like the best option (no more than 4 are allowed here since there are only 8 multinomial trials per observation and the mixture of multinomials requires $2m \leq r + 1$ for identifiability):

```
> data("Waterdata")
> cutpts <- 10.5*(-6:6)
> watermult <- makemultdata(Waterdata, cuts = cutpts)
> set.seed(10)
> multmixmodel.sel(watermult, comps = 1:4, epsilon = 0.001)

number of iterations= 44
number of iterations= 78
number of iterations= 81
               1         2         3         4 Winner
AIC    -8097.335 -3279.314 -3164.995 -3149.588      4
BIC    -8123.360 -3333.366 -3247.075 -3259.695      3
CAIC   -8129.860 -3346.866 -3267.575 -3287.195      3
ICL    -8123.360 -3332.674 -3246.042 -3258.590      3
Loglik -8084.335 -3252.314 -3123.995 -3094.588      4
```

Young (2007) gives more applications of these functions to real datasets.

6.2. Bayesian methods

Currently, there are only two **mixtools** functions relating to Bayesian methodology and they both pertain to analyzing mixtures of regressions as described in Hurn *et al.* (2003). The `regmixMH` function performs a Metropolis-Hastings algorithm for fitting a mixture of regressions model where a proper prior has been assumed. The sampler output from `regmixMH` can then be passed to `regcr` in order to construct credible regions of the regression lines. Type `help("regmixMH")` and `help("regcr")` for details and an illustrative example.

Acknowledgments

This research is partially supported by NSF Award SES-0518772. DRH received additional funding from Le Studium, an agency of the Centre National de la Recherche Scientifique of France.

References

Akaike H (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In BN Petrov, F Csaki (eds.), *Second International Symposium on Information Theory*, pp. 267–281. Akademiai Kiado.

Benaglia T, Chauveau D, Hunter DR (2009a). "An EM-Like Algorithm for Semi- and Non-Parametric Estimation in Multivariate Mixtures." *Journal of Computational and Graphical Statistics*, **18**, 505–526.

Benaglia T, Chauveau D, Hunter DR (2009b). "Bandwidth Selection in an EM-Like Algorithm for Nonparametric Multivariate Mixtures." *Technical Report hal-00353297, version 1*, HAL. URL http://hal.archives-ouvertes.fr/hal-00353297.

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

Biernacki C, Celeux G, Govaert G (2000). "Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(7), 719–725.

Bordes L, Chauveau D, Vandekerkhove P (2007). "A Stochastic EM Algorithm for a Semiparametric Mixture Model." *Computational Statistics & Data Analysis*, **51**(11), 5429–5443.

Bordes L, Mottelet S, Vandekerkhove P (2006). "Semiparametric Estimation of a Two-Component Mixture Model." *The Annals of Statistics*, **34**(3), 1204–1232.

Bozdogan H (1987). "Model Selection and Akaike's Information Criterion (AIC): The General Theory and Its Analytical Extensions." *Psychometrika*, **52**(3), 345–370.

Cruz-Medina IR, Hettmansperger TP, Thomas H (2004). "Semiparametric Mixture Models and Repeated Measures: The Multinomial Cut Point Model." *Journal of the Royal Statistical Society C*, **53**(3), 463–474.

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

Efron B, Tibshirani R (1993). *An Introduction to the Bootstrap*. Chapman & Hall, London.

Elmore RT, Hettmansperger TP, Thomas H (2004). "Estimating Component Cumulative Distribution Functions in Finite Mixture Models." *Communications in Statistics – Theory and Methods*, **33**(9), 2075–2086.

Elmore RT, Wang S (2003). "Identifiability and Estimation in Finite Mixture Models with Multinomial Coefficients." *Technical Report 03-04*, Penn State University.

Fraley C, Raftery A (2009). ***mclust**: Model-Based Clustering and Normal Mixture Modeling*. R package version 3.3.1, URL http://CRAN.R-project.org/package=mclust.

Grün B, Leisch F (2008). "FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters." *Journal of Statistical Software*, **28**(4), 1–35. URL http://www.jstatsoft.org/v28/i04/.

Hall P, Neeman A, Pakyari R, Elmore RT (2005). "Nonparametric Inference in Multivariate Mixtures." *Biometrika*, **92**(3), 667–678.

Hall P, Zhou XH (2003). "Nonparametric Estimation of Component Distributions in a Multivariate Mixture." *The Annals of Statistics*, **31**(1), 201–224.

Hettmansperger TP, Thomas H (2000). "Almost Nonparametric Inference for Repeated Measures in Mixture Models." *Journal of the Royal Statistical Society B*, **62**(4), 811–825.

Hunter DR, Wang S, Hettmansperger TP (2007). "Inference for Mixtures of Symmetric Distributions." *The Annals of Statistics*, **35**, 224–251.

Hurn M, Justel A, Robert CP (2003). "Estimating Mixtures of Regressions." *Journal of Computational and Graphical Statistics*, **12**(1), 55–79.

Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991). "Adaptive Mixtures of Local Experts." *Neural Computation*, **3**(1), 79–87.

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

McLachlan GJ (1987). "On Bootstrapping the Likelihood Ratio Test Statistic for the Number of Components in a Normal Mixture." *Journal of the Royal Statistical Society C*, **36**, 318–324.

McLachlan GJ, Peel D (2000). *Finite Mixture Models*. John Wiley & Sons, New York.

Quandt RE (1972). "The Estimation of the Parameters of a Linear Regression System Obeying Two Separate Regimes." *Journal of the American Statistical Association*, **67**(338), 306–310.

Quandt RE, Ramsey JB (1978). "Estimating Mixtures of Normal Distributions and Switching Regressions." *Journal of the American Statistical Association*, **73**(364), 730–738.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

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

Silverman BW (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC.

Young DS (2007). *A Study of Mixtures of Regressions*. Ph.D. thesis, The Pennsylvania State University. Unpublished.

Young DS, Hunter DR (2009). "Mixtures of Regressions with Predictor-Dependent Mixing Proportions." *Technical Report 09-03*, Penn State University. URL http://www.stat.psu.edu/reports/.

Affiliation:

Didier Chauveau
Laboratoire MAPMO - UMR 6628 - Fédération Denis Poisson
Université d'Orléans
BP 6759, 45067 Orléans cedex 2, FRANCE.
E-mail: didier.chauveau@univ-orleans.fr
URL: http://www.univ-orleans.fr/mapmo/membres/chauveau/

David R. Hunter
Department of Statistics
326 Thomas Building
Pennsylvania State University
University Park, PA 16802
Telephone: +1/814-863-0979
Fax: +1/814-863-7114
E-mail: dhunter@stat.psu.edu
URL: http://www.stat.psu.edu/~dhunter/

Tatiana Benaglia
Department of Statistics, Penn State (see above)
E-mail: tab321@stat.psu.edu

Derek Young
Department of Statistics, Penn State (see above)
E-mail: dsy109@psu.edu