archives-ouvertes.fr

# Bandwidth Selection in an EM-like algorithm for nonparametric multivariate mixtures

Tatiana Benaglia, Didier Chauveau, David Hunter

## ▶ To cite this version:

HAL Id: hal-00353297

https://hal.archives-ouvertes.fr/hal-00353297

Submitted on 15 Jan 2009

# Bandwidth Selection in an EM-like algorithm for nonparametric multivariate mixtures

Tatiana Benaglia[1]   Didier Chauveau[2]   David R. Hunter[1]

January 2009

[1]Department of Statistics
Pennsylvania State University, University Park, PA 16802
E-mail: `tab321@stat.psu.edu, dhunter@stat.psu.edu`

[2]MAPMO — UMR 6628 — Fédération Denis Poisson
Université d'Orléans, BP 6759, 45067 Orléans cedex 2
E-mail: `didier.chauveau@univ-orleans.fr`

## Abstract

In this paper we describe a method to select the bandwidth used in the nonparametric EM (npEM) algorithm of Benaglia et al. (2008). This method is a generalization of the Silverman's rule of thumb used to select a bandwidth in kernel density estimation, and it results in one bandwidth for each mixture component and each block of conditionally independent and identically distributed repeated measures.

**Keywords:** EM algorithm; kernel density estimation; multivariate mixture; nonparametric mixture

## 1   Introduction

Suppose the $r$-dimensional vectors $\mathbf{X}_1, \ldots, \mathbf{X}_n$ are a simple random sample from a finite mixture of $m > 1$ distributions. For nonparametric mixtures, we do *not* assume that the component distributions come from a family of densities that may be indexed by a finite-dimensional parameter vector. However, it is necessary to restrict the family $\mathcal{F}$ of multivariate density functions from which the component densities are drawn in order to avoid the problem of model non-identifiability, as discussed by Benaglia et al. (2008) and several of the references therein. To this end, we assume throughout

this article that $\mathcal{F}$ contains only densities equal to the product of their $r$ univariate marginal densities. In other words, the coordinates of the $\mathbf{X}_i$ vector are independent, conditional on the subpopulation or component (1 through $m$) from which $\mathbf{X}_i$ is drawn. This *conditional independence assumption* has appeared in a growing body of literature on non- and semi-parametric multivariate mixture models; see Benaglia et al. (2008) for a discussion of the relevant literature. Also see Allman et al. (2008) and Kasahara and Shimotsu (2008) for recent breakthroughs related to the identifiability of these models. We avoid discussion of the identifiability question here, except to state that Allman et al. (2008) give mild sufficient conditions for identifiability whenever $r \geq 3$.

We let $\boldsymbol{\theta}$ denote the vector of parameters, including the mixing proportions $\lambda_1, \ldots, \lambda_m$ and the univariate densities $f_{jk}$ (here and throughout this article, $j$ indexes the component and $k$ indexes the coordinate, so $1 \leq j \leq m$ and $1 \leq k \leq r$). Thus, under the assumption of conditional independence, the mixture density evaluated at $\mathbf{x}_i = (x_{i1}, \ldots, x_{ir})^t$ is

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

In many models, there is reason to assume that *some* or *all* of the $r$ coordinate densities $f_{j1}(\cdot), \ldots, f_{jr}(\cdot)$ are the same for all $j$. For instance, Elmore et al. (2004) and related articles assume that the coordinates are conditionally independent and identically distributed (iid), i.e., $f_{j1}(\cdot) = \cdots = f_{jr}(\cdot)$ for all $j$. In order to encompass both the conditionally iid case and the more general case simultaneously in the model and our algorithm, we allow that the coordinates of $\mathbf{X}_i$ are conditionally independent, and that there exist *blocks* of coordinates that are also identically distributed. These blocks may all be of size one so that the general case is still covered, or there may exist only a single block of size $r$, which is the conditional iid case. If we let $b_k$ denote the block to which the $k$th coordinate belongs, where $1 \leq b_k \leq B$ and $B$ is the total number of such blocks, the general model is

$$g_{\boldsymbol{\theta}}(\mathbf{x}_i) = \sum_{j=1}^{m} \lambda_j \prod_{k=1}^{r} f_{jb_k}(x_{ik}). \tag{2}$$

Benaglia et al. (2008) give an algorithm for estimating $\theta$ — i.e., the mixing weights $\lambda_j$ and the densities $f_{jb_k}(\cdot)$ — that is based on the well-known family of EM algorithms for parametric mixture models. After introducing the Benaglia et al. (2008) algorithm, this article presents several extensions of this algorithm that make it more flexible.

## 2 The nonparametric EM algorithm

The algorithm described in Benaglia et al. (2008), which is implemented as a function called "npEM" in the `mixtools` package (Young et al., 2009) for R (R Development Core Team, 2008), operates as follows: Given initial values $\boldsymbol{\theta}^0 = (\boldsymbol{\lambda}^0, \mathbf{f}^0)$, iterate the following three steps for $t = 0, 1, \ldots$:

- **E-step:** Letting $Z_{ij}$ denote the (unobserved) indicator of the event that the $i$th observation is drawn from the $j$th component, calculate the "posterior" probabilities (conditional on the data and $\boldsymbol{\theta}^t$) of component inclusion,

$$
\begin{aligned}
p_{ij}^t &\stackrel{\text{def}}{=} P_{\boldsymbol{\theta}^t}(Z_{ij} = 1 | \mathbf{x}_i) \qquad\qquad\qquad (3) \\
&= \frac{\lambda_j^t \prod_{k=1}^r f_{jb_k}^t(x_{ik})}{\sum_{j'=1}^m \lambda_{j'}^t \prod_{k=1}^r f_{j'b_k}^t(x_{ik})}, \qquad\quad (4)
\end{aligned}
$$

for all $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

- **M-step:** Set

$$
\lambda_j^{t+1} = \frac{1}{n} \sum_{i=1}^n p_{ij}^t \qquad\qquad\qquad (5)
$$

for $j = 1, \ldots, m$.

- **Nonparametric density estimation step:** For each component $j \in \{1, \ldots, m\}$ and each block $\ell \in \{1, \ldots, B\}$, define the function

$$
\begin{aligned}
f_{j\ell}^{t+1}(u) &= \frac{\frac{1}{h} \sum_{k=1}^r \sum_{i=1}^n p_{ij}^t I\{b_k = \ell\} K\left(\frac{u-x_{ik}}{h}\right)}{\sum_{k=1}^r \sum_{i=1}^n p_{ij}^t I\{b_k = \ell\}} \\
&= \frac{1}{nhC_\ell \lambda_j^{t+1}} \sum_{k=1}^r \sum_{i=1}^n p_{ij}^t I\{b_k = \ell\} K\left(\frac{u - x_{ik}}{h}\right), \quad (6)
\end{aligned}
$$

where $K(\cdot)$ is a kernel density function, $h$ is a bandwidth (see section 3), and

$$
C_\ell = \sum_{k=1}^r I\{b_k = \ell\} \qquad\qquad\qquad (7)
$$

is the number of coordinates in the $\ell$th block.

Benaglia et al. (2008) give modified versions of the npEM algorithm tailored for specific situations (iid coordinates, components differing only by a location or scale parameter, univariate symmetric components, and others). Extensions of several of these versions are discussed in Benaglia et al. (2008).

# 3   Bandwidth Selection

The nonparametric density estimation step (6) of the npEM algorithm described above is a modified version of kernel density estimation, a well-studied topic in statistics (e.g., see Silverman, 1986; Scott, 1992; Härdle et al., 2004). The central decision in this step is the selection of an appropriate value for the bandwidth $h$, or smoothing parameter, since this choice affects density estimates dramatically. The choice of kernel function $K(\cdot)$ is not as influential, so we simply take the kernel to be the standard normal density function.

There exist some standard rules in the literature for choosing the bandwidth in the univariate case when there is no mixture structure. We will use some of these standard ideas and extend them to the mixture case. However, selecting a bandwidth in a mixture setting like this one appears to be a fundamentally more complicated problem than in the corresponding non-mixture case due to the fact that we do not obtain direct information about the individual densities $f_{j\ell}$ from a mixture sample.

Benaglia et al. (2008) use a single fixed bandwidth $h$ for all components and blocks, selected by default according to a rule of thumb due to Silverman (1986, page 48). The entire $n \times r$ dataset is treated as a vector of length $nr$ and then

$$h = 0.9 \min\left\{\text{SD}, \frac{\text{IQR}}{1.34}\right\} (nr)^{-1/5}, \tag{8}$$

where SD and IQR are respectively the standard deviation and interquartile range of all $nr$ data values.

The nonparametric mixture setting makes bandwidth selection challenging, and Benaglia et al. (2008) suggest several reasons why a method like equation (8) might produce an under- or over-estimate for the bandwidth. First, pooling all of the data implicitly treats all of the different components as though they are from the same distribution. This can overestimate the bandwidth, particularly if the mixture components' centers are well-separated, because in that case, the variability of the pooled dataset will be larger than that of the individual components. Similarly, if the vector coordinates are not identically distributed within each component, the bandwidth could be biased upward for the same reason. On the other hand, note that the expression $nr$ in equation (8) is an overestimate of the "true" sample size obtained from one of the components or one of the iid coordinate blocks, especially when each of the $r$ coordinates forms its own block as in equation (1), in which case it may be sensible to eliminate the $r$ from the equation (8) entirely. But regardless of whether the coordinates are combined in blocks or not, the fact remains that the "true" sample size from each component is actually some fraction of $n$, namely, about $n\lambda_j$ for the $j$th component.

A further limitation of the Benaglia et al. (2008) algorithm is that equation (6) forces each component and block to employ the same bandwidth,

when in fact there is no reason to assume that these bandwidths should be the same. In other words, we may simply modify the algorithm from Benaglia et al. (2008) by replacing $h$ by $h_{j\ell}$ in equation (6), thus allowing a different value of the bandwidth for each component and each block density function. The remainder of this section describes how to modify the Benaglia et al. (2008) algorithm to allow for this greater flexibility.

As noted earlier, the challenge of bandwidth selection for a mixture model is that a sample from $g_{\boldsymbol{\theta}}(\mathbf{x})$ does not give direct information about the individual densities $f_{j\ell}$. However, after each iteration of the npEM algorithm, we do have density estimates $f_{j\ell}^{t+1}$. This suggests an iterative scheme whereby re-estimation of the $h_{j\ell}$ values becomes part of a modified npEM algorithm at each iteration.

To adapt Silverman's rule of thumb (Silverman, 1986, page 48) to select the value of $h_{j\ell}$ in an iterative procedure, for each iteration of the npEM algorithm, we need an estimate of the sample size, the sample standard deviation, and the interquartile range for each component and each block. Once these estimates are in place, the estimated bandwidths at the $(t+1)$th iteration, calculated just before the density estimation step of the algorithm, are given by:

$$h_{j\ell}^{t+1} = 0.9 \min\left\{ \sigma_{j\ell}^{t+1}, \frac{IQR_{j\ell}^{t+1}}{1.34} \right\} (nC_\ell \lambda_j^{t+1})^{-1/5}, \qquad (9)$$

where $nC_\ell \lambda_j^{t+1}$ estimates the sample size for the $\ell$th block of coordinates in the $j$th component, and $\sigma_{j\ell}^{t+1}$ and $IQR_{j\ell}^{t+1}$ are the weighted standard deviation and empirical interquartile range for the $j$th component and $\ell$th block. Calculation of $\sigma_{j\ell}^{t+1}$ is fairly straightforward if we augment each M-step to include

$$\mu_{j\ell}^{t+1} = \frac{\displaystyle\sum_{i=1}^{n}\sum_{k=1}^{r} p_{ij}^t I\{b_k = \ell\} x_{ik}}{\displaystyle\sum_{i=1}^{n}\sum_{k=1}^{r} p_{ij}^t I\{b_k = \ell\}} = \frac{\displaystyle\sum_{i=1}^{n}\sum_{k=1}^{r} p_{ij}^t I\{b_k = \ell\} x_{ik}}{n\lambda_j^{t+1} C_\ell} \qquad (10)$$

and

$$\sigma_{j\ell}^{t+1} = \left[ \frac{1}{nC_\ell \lambda_j^{t+1}} \sum_{i=1}^{n}\sum_{k=1}^{r} p_{ij}^t I\{b_k = \ell\}(x_{ik} - \mu_{j\ell}^{t+1})^2 \right]^{1/2}. \qquad (11)$$

We compute $IQR_{j\ell}^{t+1}$ as the difference between the estimated 0.75 and 0.25 quantiles of the $\ell$th block of the $j$th component. To accomplish this, we first introduce the notion of a weighted quantile estimate:

Let $a_1, \ldots, a_\nu$ be real numbers and $w_1, \ldots, w_\nu$ be associated (nonnegative) weights, with $W = w_1 + \cdots + w_\nu$. The first step in finding the weighted

quantile estimate is to sort the $a_i$ in non-decreasing order. To this end, let $\tau(\cdot)$ be a permutation on the integers $\{1, \ldots, \nu\}$ such that

$$a_{\tau(1)} \leq a_{\tau(2)} \leq \cdots \leq a_{\tau(\nu)}.$$

(The $\tau$ permutation need not be unique if there are ties among the $a_i$.) Then for $\alpha \in (0, 1)$, we define the weighted $\alpha$ quantile estimate to be $a_{\tau(i_\alpha)}$, where

$$i_\alpha = \min \left\{ s : \sum_{i=1}^{s} w_{\tau(i)} \geq \alpha W \right\}$$

is the smallest integer that gives at least a proportion $\alpha$ of the total sum of weights $W$. Note that in the special case in which all $w_i$ are the same, the weighted quantile estimate is simply a particular way to define the regular sample quantile. The `mixtools` package includes a `wquantile` function, based on the `findInterval` function in R, that implements finding the weighted quantile.

To find $IQR_{j\ell}^{t+1}$, we first calculate the weighted 0.25 and 0.75 quantile estimate of the $nC_\ell$ data values in block $\ell$, with corresponding weights given by the posterior probabilities $p_{ij}^t$. The weighted interquartile range is then the difference between these two quantiles. In the `mixtools` package, the function `wIQR` calculates the weighted interquartile range. Note that when this calculation is performed as part of the npEM algorithm, the permutation $\tau$ need only be calculated once due to the fact that the data and block structure do not change during the running of the algorithm. Furthermore, the sum $W$ of the weights at iteration $t$ is equal to $\lambda_j^{t+1} C_\ell$ because of equation (5).

Note that there are alternatives to the bandwidth selection method of Silverman (1986) given by equation (9). For instance, simply replacing 0.9 by 1.06 in equation (9) yields the method of Scott (1992). Further bandwidth selection methods are detailed in the documentation for the `bw.nrd0` function in R (R Development Core Team, 2008), and these might be adapted to the case of a weighted sample just as the Silverman (1986) and Scott (1992) methods have been.

## 4  An example

A simple example serves to illustrate the effect of allowing different bandwidths in the npEM algorithm. We generated 300 observations from a $m = 2$ component mixture from the general model (2) with trivariate ($r = 3$) observations, $B = 1$ block (i.e. $b_1 = b_2 = b_3 = 1$, which means that we have three conditionally iid repeated measures), and parameter and densities:

$$\lambda_1 = 0.4, \quad f_{11} \equiv \mathcal{N}(0, 1), \quad f_{21} \equiv \mathcal{N}(15, 25). \tag{12}$$

We estimate the model applying the npEM algorithm using both a single bandwidth given by (8) and different bandwidths computed by (9). The code used to generate the data and then apply the npEM algorithm in R using the `mixtools` package may be found in the appendix. Note that in the actual npEM output, the component labels were switched — i.e., actually $\hat{h}_{11} = 1.246$ and $\hat{h}_{21} = 0.266$ — but we have relabeled the components to agree with the definitions of equation (12).

If we consider each component separately and apply Silverman's rule of thumb, the bandwidths would be $h_1 = 0.9(0.4 \times 300)^{-1/5} \approx 0.35$, and $h_2 = 0.9 \times 5 \times (0.6 \times 300)^{-1/5} \approx 1.593$, for the first and second component respectively. The npEM algorithm used $\hat{h} = 1.932$ for a single bandwidth, and $\hat{h}_{11} = 0.266$ and $\hat{h}_{21} = 1.246$ when allowing different bandwidths. The left-hand plot of Figure 1 shows that the large value $\hat{h}$ results in an over-smoothed density estimate for the first component. Allowing separate bandwidth estimates $\hat{h}_{11}$ and $\hat{h}_{21}$, as in the right-hand plot of Figure 1, gives better results.
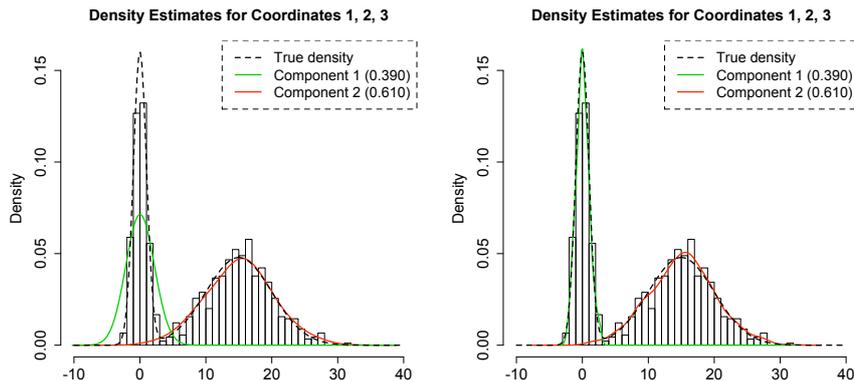


Figure 1: Effect of bandwidth in the component density estimates. On the left are estimates from the original npEM algorithm using the same bandwidths; on the right are estimates using the modified npEM algorithm in which individual component bandwidths are updated at each iteration.

In Figure 1, we see from the legends that the final estimates of $\lambda_1$ and $\lambda_2$ are the same for the two algorithms, i.e., $\hat{\lambda}_1 = 0.390$ and $\hat{\lambda}_2 = 0.610$. This is not guaranteed to be the case: Since the $\lambda$ estimates are ultimately functions of the density estimates in the npEM algorithm, the original (fixed, single $h$) npEM algorithm can lead to different estimates that the modified algorithm we introduce here. However, the particular example we have chosen involves mixture components that are "well-separated" in the sense that it is fairly easy to assign each observation to one component or the other. Thus, the final estimate of $\lambda_1$ is simply the proportion of observations that are classified

as belonging to the first component, which is 117/300 in this example. We may observe that this is true by noting that all of the posterior probabilities $p_{ij}$ at the final iteration are very close to zero or one. (In fact, the smallest value of $\max_j p_{ij}$ is larger than 0.9999 for both algorithms of Figure 1.)

## 5 Further extensions

Among the special cases of the npEM algorithm discussed by Benaglia et al. (2008) are those in which component or block density estimates may differ from one another, but they differ only in (say) a location or scale parameter and otherwise they have the same shape. In cases such as these, an extension of the npEM algorithm as we have suggested here — recalculating each $h_{jk}$ value at each iteration — should be designed to exploit this structure.

For instance, the most restrictive case discussed in Benaglia et al. (2008) is the case in which the density function for each component and block shares exactly the same shape, and the different components and blocks differ only by a possible location and scale parameter, so that for every component $j$ and block $\ell$, the $(j, \ell)$ density function becomes

$$f_{j\ell}(x) \equiv \frac{1}{\sigma_{j\ell}} f\left(\frac{x - \mu_{j\ell}}{\sigma_{j\ell}}\right). \tag{13}$$

In this case, because we wish to estimate only a single density $f(x)$, the density estimation step in the algorithm would specify for any $u \in \mathbb{R}$ that

$$f^{t+1}(u) = \frac{1}{nrh} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{r} p_{ij}^t K\left(\frac{u - x_{ik} + \mu_{jb_k}}{h\sigma_{jb_k}}\right). \tag{14}$$

In particular, note that there is no reason to allow a different bandwidth $h_{j\ell}$ for each component $j$ and block $\ell$ as in equation (9). In effect, the different bandwidths for each component and block have been entirely accounted for by the location and scale parameters. Indeed, in the example of Section 4, the component and block densities all have exactly the same true shape — normal — so that by accounting for the location and scale changes, we eliminate the need in that example for different bandwidths. Even in this case, however, the updating of $h$ at each iteration as in equation (9) is still useful, since it is still difficult for the simplistic bandwidth-selection rule of thumb employed by Benaglia et al. (2008) to make a good selection before anything is known about the mixture structure.

To update $h$ at each iteration under model (13), we cannot merely use a slightly modified version of equation (9) since the $\mu_{j\ell}$ and $\sigma_{j\ell}$ parameters now play a more integral role in the model and indeed the whole point of iteratively updating $h$ is to somehow ignore the location and scale parameters for each component and block. In other words, we should apply the bandwidth-selection rule of thumb only *after* each data point $x_{ik}$ has been appropriately

shifted and scaled. Since each $x_{ik}$ will actually be shifted and scaled according to *each* of the $m$ components, naturally each $(x_{ik} - \mu_{jb_k})/\sigma_{jb_k}$ must be weighted according to the corresponding weight $p_{ij}$. Since the sum of all of these weights is exactly $n$, the effective sample size is $nr$ and thus we obtain the following update for $h$:

$$h^{t+1} = \frac{0.9}{(nr)^{1/5}} \times \min\left\{1, \frac{IQR^{t+1}}{1.34}\right\},\tag{15}$$

where $IQR^{t+1}$ is the weighted interquartile range of the $n \times m \times r$ values $(x_{ik} - \mu_{jb_k})/\sigma_{jb_k}$, with corresponding weights $p_{ij}^t$, for all $1 \leq i \leq n$, $1 \leq j \leq m$, and $1 \leq k \leq r$. Note that the weighted mean and weighted variance of these same values are 0 and 1 due to the fact that each $x_{ik}$ is properly normalized within each of the $m$ components, which is why the standard deviation of equation (9) has been simply replaced by one in equation (15).
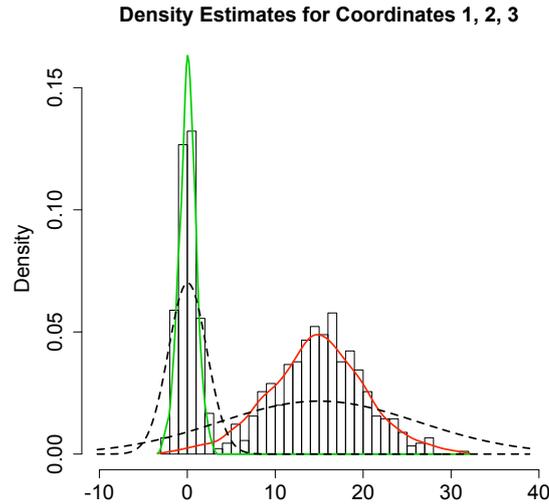


Figure 2: Two algorithms fitting model (13) are compared, one in which the bandwidth is fixed throughout (dashed lines) and one in which the bandwidth is updated at each iteration (solid lines). For each of these two methods, the two components' density estimates are identical in shape and differ only in location and scale.

In Figure 2, we can see the effect of iteratively updating the bandwidth. Each of the two algorithms whose output is displayed in that figure started at the same default bandwidth value of 1.932, yet the iteratively updated bandwidth decreased dramatically at the second iteration, finally settling on a final value of 0.220. The dashed lines show that the algorithm that fails to update this bandwidth produces a far poorer density estimate than the

one that does the updating. This is not surprising, since failing to update means that the bandwidth must be chosen prior to knowing anything about the mixture structure, at which point it is very difficult to properly choose a bandwidth.

For some applications, model (13) may be too restrictive in its assumption that all components and blocks have exactly the same shape of density function. As Benaglia et al. (2008) discuss, it is also possible to allow each component or each block to have its own density shape, which leads to replacing (13) by

$$
f_{j\ell}(x) \equiv \frac{1}{\sigma_{j\ell}} f_j \left( \frac{x - \mu_{j\ell}}{\sigma_{j\ell}} \right) \qquad \text{or} \qquad f_{j\ell}(x) \equiv \frac{1}{\sigma_{j\ell}} f_\ell \left( \frac{x - \mu_{j\ell}}{\sigma_{j\ell}} \right),
$$

respectively. Implementation of our algorithm for either of these models is relatively straightforward using the same techniques described earlier; and, in either case, the bandwidth ($h_j$ or $h_\ell$) may once again be allowed to depend upon either the component or the block, as the case may be.

# 6   Summary

This article extends the simplistic bandwidth-selection scheme of Benaglia et al. (2008) by allowing for (1) iteratively updating bandwidths; and (2) component- and block-specific bandwidths. Each of these two extensions improves the density estimations in certain cases for a different reason: The first solves the problem that it is difficult to estimate a bandwidth before knowing about the mixture structure, while the second takes care of cases in which different components or blocks have very different properties. In cases where "very different properties" includes *only* location and/or scale differences, it is possible (as we explain in Section 5) to correct for these differences in the model without component- and block-specific bandwidths; yet in more general cases where these different bandwidths are desirable, implementation of common bandwidth selection methods such as those of Scott (1992) or Silverman (1986) requires weighted versions of the interquartile range and the standard deviation as we present here.

# References

Allman, E. S., Matias, C., and Rhodes, J. A. (2008). Identifiability of latent class models with many observed variables. Technical Report 0809.5032v1, arxiv.org.

Benaglia, T., Chauveau, D., and Hunter, D. R. (2008). An EM-like algorithm for semi-and non-parametric estimation in multivariate mixtures. Technical Report hal-00193730, hal.archives-ouvertes.fr.

Elmore, R. T., Hettmansperger, T. P., and Thomas, H. (2004). Estimating component cumulative distribution functions in finite mixture models. *Comm. Statist. Theory Methods*, 33(9):2075–2086.

Härdle, W., Müller, M., Sperlich, S., and Werwatz, A. (2004). *Nonparametric and semiparametric models*. Springer Series in Statistics. Springer-Verlag, New York.

Kasahara, H. and Shimotsu, K. (2008). Nonparametric identification and estimation of multivariate mixtures. Technical Report 1153, Queen's University Department of Economics.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Scott, D. W. (1992). *Multivariate density estimation*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons Inc., New York. Theory, practice, and visualization, A Wiley-Interscience Publication.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Monographs on Statistics and Applied Probability. Chapman & Hall, London.

Young, D. S., Benaglia, T., Chauveau, D., Elmore, R. T., Hettmansperger, T. P., Hunter, D. R., Thomas, H., and Xuan, F. (2009). mixtools: Tools for mixture models. R package version 0.3.3.

### Appendix: Computer code

The following computer code can be used in R, once `mixtools` version 0.3.3 is installed, to produce the examples seen in this article.

```
library(mixtools) # mixtools package must be installed first
set.seed(123) # Ensure that results are exactly reproducible

#Generate data:
mu <- matrix(c(0, 15), 2, 3)
sigma <- matrix(c(1, 5), 2, 3)
x <- rmvnormmix(300, lambda = c(.4,.6), mu = mu, sigma = sigma)

# npEM algorithm results:
a <- npEM(x, mu0 = 2, blockid = rep(1,3), samebw = TRUE)
b <- npEM(x, mu0 = 2, blockid = rep(1,3), samebw = FALSE)

# Produce plots like in Figure 1:
```

```
s <- seq(-10, 40, len = 200)
plot(a, xlim=c(-10, 40), ylim = c(0, .16), xlab = "", breaks = 30)
lines(s, dnorm(s)*.4 + dnorm((s-15)/5)/5*.6, lwd = 2, lty = 2)
plot(b, xlim=c(-10, 40), ylim = c(0, .16), xlab = "", breaks = 30)
lines(s, dnorm(s)*.4 + dnorm((s-15)/5)/5*.6, lwd = 2, lty = 2)

# Display npEM bandwidths, minimum values of max_j p_{ij}
a$bandwidth
b$bandwidth
min(apply(a$posteriors, 1, max))
min(apply(b$posteriors, 1, max))

# spEM algorithm results:
d <- spEM(x, mu0 = 2, blockid = rep(1,3), constbw = FALSE)
d2 <- spEM(x, mu0 = 2, blockid = rep(1,3), constbw = TRUE)

# Produce plot like in Figure 2:
plot(d, xlim=c(-10, 40), ylim = c(0, .16), xlab = "", breaks = 30,
     addlegend=FALSE)
plot(d2, newplot=FALSE, addlegend=FALSE, lty=2, dens.col=1)

# Display spEM bandwidths
d$bandwidth
d2$bandwidth
```