Layered and chained mixture models with the `lcmix` package (version 0.1)

Daniel Dvorkin

August 2, 2011

The `lcmix` package fits layered and chained mixture models, which are special cases of Bayesian belief networks, to lists of data. It also offers mixture models for single data sources, similarly to the `mixtools` (Benaglia et al., 2009) and `mclust` (Fraley and Raftery, 2002, 2006, revised 2009) packages, but offers a wider choice of distributions; currently univariate and multivariate normal (Gaussian), Weibull, and gamma distributions are supported, with more distributions to be added in future releases. The primary tool used in model fitting is the EM algorithm as described in McLachlan and Krishnan (2008).

The following sections describe the models used, how to fit models, ways to evaluate the quality of the fitted models, and procedures for simulating data. Users seeking a quick start guide can proceed directly to §2, "Fitting models to data," on p. 5. However, it is strongly recommended to be familiar with the material in §1 to understand the behavior of the model fitting algorithms.

Contents

1 Model specification

The basic idea of hidden-variable mixture models such as those used in `lcmix` is that the "complete data" consists of the observed data and some hidden data, consisting of discrete components which generate the distribution of the observed data. The following subsections describe single-data mixture models, distributions for observed data, and finally the multiple-data (layered and chained) mixture models which are the focus of `lcmix`.

1.1 Single-data mixture models

In the simple single-data finite mixture model, there exists a hidden categorical random variable (RV) $\mathcal{Y}$ taking on values from 1 to some positive integer K, and having the probability density function (PDF)

$$\mathrm{f}(y) \;=\; \prod_k p_k^{\mathrm{I}(y=k)} \;=\; p_y \tag{1}$$

where $\mathrm{I}(\cdot)$ is the indicator function, $k = 1, \ldots, K$, and $\boldsymbol{p} = (p_1, \ldots, p_k)$ is a vector of probabilities such that $\sum_k p_k = 1$. That is, $p_k = \Pr(\mathcal{Y} = k)$.

Then $\mathcal{Y}$ generates the (possibly multivariate) observed RV $\boldsymbol{\mathcal{X}}$ as follows. Let $\mathrm{f}(\boldsymbol{x} \mid \theta)$ be a PDF on the sample space of $\boldsymbol{\mathcal{X}}$ with parameters θ, and let θ_y denote a particular set of parameters specified by $\mathcal{Y} = y$. Then the conditional PDF of $\boldsymbol{\mathcal{X}}$ is

$$\mathrm{f}(\boldsymbol{x} \mid \mathcal{Y} = y) \;=\; \prod_k \mathrm{f}(\boldsymbol{x} \mid \theta_k)^{\mathrm{I}(y=k)} \;=\; \mathrm{f}(\boldsymbol{x} \mid \theta_y) \tag{2}$$

from which it follows that the complete-data PDF is

$$\mathrm{f}(\boldsymbol{x}, y) \;=\; \prod_k \left\{p_k \; \mathrm{f}(\boldsymbol{x} \mid \theta_k)\right\}^{\mathrm{I}(y=k)} \;=\; p_y \; \mathrm{f}(\boldsymbol{x} \mid \theta_y) \tag{3}$$

and the marginal PDF is

$$\mathrm{f}(\boldsymbol{x}) = \sum_k p_k \; \mathrm{f}(\boldsymbol{x} \mid \theta_k). \tag{4}$$

1.2 Distributions for observed data

The univariate observed data distributions supported in the current release of `lcmix` are the normal, the Weibull (shape-decay parameterization), and the gamma. The PDF of the normal distribution with parameters $\theta = (\mu, \sigma)$, where μ is the mean and $\sigma > 0$ is the standard deviation, is

$$\mathrm{f}(x \mid \theta) = \tfrac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\tfrac{(x-\mu)^2}{2\sigma^2}\right\} \tag{5}$$

Note that the expression in Equation (5) is often written as $\phi(x \mid \theta)$, or for the standard normal PDF, that is, the PDF for $\mu = 0$, $\sigma = 1$, simply $\phi(x)$. Similarly, the standard normal CDF is often denoted by $\Phi(x)$, and its inverse by $\Phi^{-1}(u)$, $u \in [0, 1]$. The parameters of the distribution of the observed variable in a single-data normal mixture model are $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_K)$, with $\theta_y = (\mu_y, \sigma_y)$.

The shape-decay parameterization of the Weibull distribution, or the "WeiSD" distribution, with shape parameter $\sigma > 0$ and decay parameter $\delta > 0$ such that $\theta = (\sigma, \delta)$, has the PDF

$$\mathrm{f}(x \mid \theta) = \sigma\delta x^{\sigma-1} \exp(-\delta x^{\sigma}). \tag{6}$$

Compared to the usual R shape-scale parameterization, with shape parameter α and scale parameter β, the relationship between the parameters is given by $\alpha = \sigma$ and $\beta = \delta^{-1/\sigma}$, or equivalently, $\sigma = \alpha$ and $\delta = \beta^{-\alpha}$. The CDF has the very simple form $\mathrm{F}(x \mid \theta) = \exp(-\delta x^{\sigma})$. The observed variable distribution parameters in a single-data mixture model are $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_K)$ and $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_K)$, with $\theta_y = (\sigma_y, \delta_y)$.

The gamma distribution with shape parameter $\sigma > 0$ and rate parameter $\lambda > 0$ such that $\theta = (\sigma, \lambda)$ has the PDF

$$\mathrm{f}(x \mid \theta) = \tfrac{\lambda^{\sigma}}{\Gamma(\sigma)} x^{\sigma-1} \exp(-\lambda x) \tag{7}$$

where $\Gamma(\cdot)$ is the gamma function. The observed variable distribution parameters in a single-data mixture model are $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_K)$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$, with $\theta_y = (\sigma_y, \lambda_y)$.

For multivariate data $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_D)$ for some positive integer D, the basic distribution is the multivariate normal, with $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is a vector of length D and $\boldsymbol{\Sigma}$ is a $D \times D$ positive definite matrix. The PDF is

$$\mathrm{f}(\boldsymbol{x} \mid \theta) = \tfrac{1}{|\boldsymbol{\Sigma}|^{1/2}(2\pi)^{D/2}} \exp\left\{-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right\} \tag{8}$$

where $|\cdot|$ denotes the determinant. The expression in Equation (8) is often written as $\phi_D(\boldsymbol{x} \mid \theta)$, with the corresponding CDF $\Phi_D(\boldsymbol{x} \mid \theta)$. Mixture model parameters are the $K \times D$ matrix $\boldsymbol{\mu}$ of which the kth row is $\boldsymbol{\mu}_k = (\mu_{k,1}, \ldots, \mu_{k,D})$, and the list $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K)$, each $\boldsymbol{\Sigma}_k$ being a $D \times D$ positive definite matrix.

A useful special case of the multivariate normal is the multivariate standard normal (MVSN), that is, a multivariate normal distribution in which the marginal distributions are all standard normal. Let $\boldsymbol{\mathcal{S}} = (\mathcal{S}_1, \ldots, \mathcal{S}_D)$ where each $\mathcal{S}_d$ is standard normal. The correlation matrix $\boldsymbol{\rho}$ is a $D \times D$ positive definite matrix of which each diagonal element is equal to 1, and the PDF of $\boldsymbol{\mathcal{S}}$ is denoted by

$$\phi_{\boldsymbol{\rho}}(\boldsymbol{s}) = \tfrac{1}{|\boldsymbol{\rho}|^{1/2}(2\pi)^{D/2}} \exp\left\{-\tfrac{1}{2}\boldsymbol{s}^{\mathrm{T}} \boldsymbol{\rho}^{-1} \boldsymbol{s}\right\} \tag{9}$$

with the corresponding CDF $\Phi_{\boldsymbol{\rho}}(s)$.

This distribution is used to construct non-normal multivariate distributions by the copula method, as described in Song (2000). Briefly, let the marginal PDF of $\mathcal{X}_d$ be denoted $\mathrm{f}(x_d \mid \theta)$, with $\mathrm{f}(\cdot)$ being one of the univariate PDF's given above, and $\mathrm{F}(x_d \mid \theta)$ be the corresponding CDF. Now let $s_d = \Phi^{-1}\{\mathrm{F}(x_d \mid \theta)\}$, and let the joint CDF for the $\mathcal{X}_d$'s be given by $\mathrm{F}(\boldsymbol{x} \mid \theta) = \Phi_{\boldsymbol{\rho}}(\boldsymbol{s})$. It follows that the joint PDF is

$$\mathrm{f}(\boldsymbol{x} \mid \theta) = \phi_{\boldsymbol{\rho}}(\boldsymbol{s}) \prod_d \tfrac{\mathrm{f}(x_d \mid \theta_d)}{\phi(s_d)} \tag{10}$$

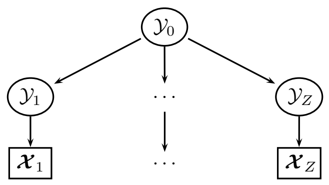

Figure 1: A layered mixture model. Ovals indicate hidden variables, while rectangles indicate observed variables. Arrows show generative relationships, which account for all dependencies in the model.

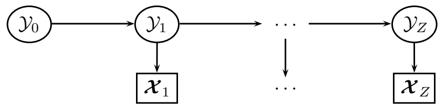

Figure 2: A chained mixture model. Ovals indicate hidden variables, while rectangles indicate observed variables. Arrows show generative relationships, which account for all dependencies in the model.

with each θ_d being the parameters for the marginal distribution of $\mathcal{X}_d$, such as $\theta_d = (\sigma_d, \delta_d)$ for the WeiSD or $\theta_d = (\sigma_d, \lambda_d)$ for the gamma. In the mixture model case, we have the parameters $\boldsymbol{\rho} = (\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K)$ and the set of $\theta_{y,d}$'s, with $\theta_{y,d}$ being the parameters of the marginal distribution of $\mathcal{X}_d$ given that $\mathcal{Y} = y$.

1.3 Multiple-data models

In the layered mixture model, there exists a single top-level hidden categorical RV $\mathcal{Y}_0$ which generates the "layer" of hidden categorical variables $\boldsymbol{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_Z)$ for some positive integer Z. The $\mathcal{Y}_z$'s, for $z = 1, \ldots, Z$, in turn generate the observed variables $\boldsymbol{\mathcal{X}} = (\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_Z)$. This model is shown in Figure 1.

In the chained mixture model, $\mathcal{Y}_0$ generates $\mathcal{Y}_1$, which in turn generates $\mathcal{Y}_2$, etc. up to $\mathcal{Y}_Z$. As in the layered model, each $\mathcal{Y}_z$ also generates $\boldsymbol{\mathcal{X}}_z$. This model is shown in Figure 2.

In both models, $\mathcal{Y}_0$ may take on values from 1 to some positive integer K_0. Its distribution is parameterized by the probability vector $\boldsymbol{p}_0 = (p_{0:1}, \ldots, p_{0:K_0})$. Each $\mathcal{Y}_z$ may take on values from 1 to some positive integer K_z, and its distribution is parameterized by the matrix $\boldsymbol{Q}_z$. In the layered model, $\boldsymbol{Q}_z$ is a $K_0 \times K_z$ matrix of which the (k_0, k_z)th element is $q_{z:k_0,k_z} = \Pr(\mathcal{Y}_z = k_z \mid \mathcal{Y}_0 = k_0)$. In the chained model, $\boldsymbol{Q}_z$ is a $K_{z-1} \times K_z$ matrix of which the (k_{z-1}, k_z)th element is $q_{z:k_{z-1},k_z} = \Pr(\mathcal{Y}_z = k_z \mid \mathcal{Y}_{z-1} = k_{z-1})$. Marginally, $\mathcal{Y}_z$ has the categorical distribution with probability vector $\boldsymbol{p}_z = \boldsymbol{Q}_z^{\mathrm{T}} \boldsymbol{p}_0$ in the layered model, or the recursively defined $\boldsymbol{p}_z = \boldsymbol{Q}_z^{\mathrm{T}} \boldsymbol{p}_{z-1}$ in the chained model.

The relationship between $\boldsymbol{\mathcal{X}}_z$ and $\mathcal{Y}_z$ in the multiple-data models is the same as that between $\boldsymbol{\mathcal{X}}$ and $\mathcal{Y}$ in the single-data model. Each $\boldsymbol{\mathcal{X}}_z$ may have any one of the distributions, conditional on $\mathcal{Y}_z$, described in §1.2. Given $\mathcal{Y}_z = y_z$, the PDF for this distribution is denoted $\mathrm{f}(\boldsymbol{x}_z \mid \theta_{z:y_z})$. Let $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Z)$ in this context, and the complete-data PDF for the layered model is most simply written as

$$\mathrm{f}(\boldsymbol{X}, \boldsymbol{y}, y_0) = p_{0:y_0} \prod_z q_{z:y_0,y_z} \mathrm{f}(\boldsymbol{x}_z \mid \theta_{z:y_z}) \tag{11}$$

with the marginal PDF

$$\mathrm{f}(\boldsymbol{X}) = \sum_{k_0} p_{0:k_0} \prod_z \sum_{k_z} q_{z:k_0,k_z} \mathrm{f}(\boldsymbol{x}_z \mid \theta_{z:k_z}). \tag{12}$$

For the chained model,

$$\mathrm{f}(\boldsymbol{X}, \boldsymbol{y}, y_0) = p_{0:y_0} \prod_z q_{z:y_{z-1},y_z} \mathrm{f}(\boldsymbol{x}_z \mid \theta_{z:y_z}). \tag{13}$$

The expression for $\mathrm{f}(\boldsymbol{X})$ is somewhat more complicated than in the layered model. The function $\alpha_z(y_z) = \mathrm{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_z, y_z)$ is recursively defined by

$$\alpha_1(y_1) = p_{1:y_1}\ \mathrm{f}(\boldsymbol{x}_1 \mid \theta_{1:y_1})$$

and for $z > 1$,

$$\alpha_z y_z = \mathrm{f}(\boldsymbol{x}_z \mid \theta_{z:y_z}) \sum_{k_{z-1}} \alpha_{z-1}(k_{z-1}) q_{z:k_{z-1},k_z}.$$

Thus the marginal PDF is

$$\mathrm{f}(\boldsymbol{X}) = \sum_{k_Z} \alpha_Z(k_Z). \tag{14}$$

See McLachlan and Krishnan (2008) pp. 290-293 for background and further details.

2 Fitting models to data

The key functions for estimating the parameters of the models described in §1 are `mixmod` for single-data models and `mdmixmod` for multiple-data models. The use of these functions is illustrated with four example data sets, `exampleData1` through `exampleData4`. For the details on the generation of these data sets, see §4, "Simulating data."

2.1 Single-data models

To begin, we load the `lcmix` package and the specified data sets:

```
> library(lcmix)
> data(exampleData1)
> data(exampleData2)
> data(exampleData3)
> data(exampleData4)
```

The `exampleData1` set contains data generated from a very simple single-data model using a univariate normal mixture. It is a list of two elements: the vector `X` of length $N = 1000$, representing the observed data; and the vector `Y`, also of length N, representing the hidden components. For model fitting, of course, we assume the components are unknown, and so `X` is treated as "the data" for input. We fit the normal mixture model with $K = 2$ components, observing that `family="normal"` is the default distribution family and so need not be specified:

```
> mod1 <- mixmod(exampleData1$X, K = 2)
```

and then print the model for examination:

```
> mod1

Normal mixture model ('norm')
Data 'exampleData1$X' of length 1000 fitted to 2 components
Model statistics:
     iter      llik      qval       bic
   44.000 -1771.990 -1828.620 -3578.518
```

The `print` method for objects of class `mixmod` tells us the distribution used in model fitting, here `"norm"`, the univariate normal; the name and size of the data; the number of components; and model statistics including the number of iterations, the log-likelihood for the parameters of the model given the observed data, the Q-value (the expected complete data log-likelihood), and the BIC (Bayesian Information Criterion) value. A `mixmod` object is actually a list with a number of elements, of which `params`, the parameters of the fitted model, are of particular interest:

```
> names(mod1)

 [1] "N"                "D"                "K"                "X"
 [5] "npar"             "npar.hidden"      "npar.observed"    "iter"
 [9] "params"           "stats"            "weights"          "pdfs"
[13] "posterior"        "assignment"       "iteration.params" "iteration.stats"
[17] "family"           "distn"            "iter.max"         "dname"
[21] "dattr"

> mod1$params

$hidden
$hidden$prob
[1] 0.09250383 0.90749617


$observed
$observed$mean
[1]  2.823488 -2.983441

$observed$var
[1] 7.329986 1.020609
```

```
$observed$sd
[1] 2.707395 1.010252
```

The `hidden` element of `params` is a list containing the estimated parameters of the distribution of the hidden components; in the single-data model it consists only of the vector `prob`, corresponding to the estimated $\hat{\boldsymbol{p}}$ parameter of the categorical distribution. The `observed` element of `params` is a list containing the estimated parameters of the distribution of the observed data; here `mean` and `sd` correspond to the $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ parameters of the univariate normal mixture distribution, along with `var` representing $\hat{\boldsymbol{\sigma}}^2 = (\hat{\sigma}_1^2, \ldots, \hat{\sigma}_K^2)$.

Also of interest are the posterior probabilities that the observed data were generated by one or the other component, given in `posterior`, an $N \times K$ matrix with elements between 0 and 1 and rows summing to 1, of which the (n, k)th element is the posterior probability that the nth datum was generated by the kth component. Since here, the components are known, we can look at part of the `posterior` matrix in comparison to the actual components and see how well the model predicts the component which generated each observed datum:

```
> cbind(mod1$posterior, component = exampleData1$Y)[1:10, ]

                               component
 [1,] 0.009657448 0.990342552         2
 [2,] 0.003108806 0.996891194         2
 [3,] 0.006163714 0.993836286         2
 [4,] 0.002681880 0.997318120         2
 [5,] 0.993942315 0.006057685         1
 [6,] 0.003420525 0.996579475         2
 [7,] 0.004054660 0.995945340         2
 [8,] 0.016711469 0.983288531         2
 [9,] 0.064393471 0.935606529         2
[10,] 0.013923883 0.986076117         2
```

To reiterate the point that the hidden data are not needed for model fitting — and, for most real-world data sets, will not in fact be available — consider the following simple example of simulating, and fitting a model to, a data set with characteristics similar to those of the `X` element of `exampleData1`:

```
> MyData <- sample(c(rnorm(100, 3, 3), rnorm(900, -3, 1)))
> MyMod <- mixmod(MyData, 2)
> MyMod

Normal mixture model ('norm')
Data 'MyData' of length 1000 fitted to 2 components
Model statistics:
     iter      llik      qval       bic
  104.000 -1777.540 -1853.454 -3589.618
```

```
> MyMod$params

$hidden
$hidden$prob
[1] 0.09749727 0.90250273


$observed
$observed$mean
[1]  2.762205 -2.990282

$observed$var
[1] 9.378092 1.013045

$observed$sd
[1] 3.062367 1.006501
```

Here, although the components are unknown thanks to the use of `sample` in creating the data, the mixing proportions and observed variable distribution parameters are estimated. The components will, of course, be guessed based on `posterior`; another element of the `mixmod` list, the vector `assignment`, gives the most probable choices.

The `exampleData2` set contains data generated from a single-data model using a multivariate WeiSD mixture. As with `exampleData1`, it is a list with two elements, `X` and `Y`, containing the observed and hidden data respectively. However, here `X` is a $N \times D$ matrix ($D = 4$). We fit a mixture model with $K = 3$ components using the `weibull` distribution family. Note that partial matches in family names are allowed, and also that `mixmod` detects that the observed data are multivariate and chooses a distribution accordingly:

```
> mod2 <- mixmod(exampleData2$X, 3, family = "wei")
> mod2

Weibull mixture model ('mvweisd')
Data 'exampleData2$X' of size 1000-by-4 fitted to 3 components
Model statistics:
    iter     llik     qval      bic
  91.000 2905.283 2328.488 5506.626

> mod2$params

$hidden
$hidden$prob
[1] 0.2648161 0.5338918 0.2012921


$observed
```

```
$observed$shape
         [,1]     [,2]     [,3]     [,4]
[1,] 11.66744 7.766927 4.630244 2.820865
[2,] 11.70439 8.136748 5.548090 1.991357
[3,]  8.46983 8.479384 3.803052 1.070830

$observed$decay
         [,1]     [,2]     [,3]     [,4]
[1,] 1.383104 4.984691 6.503916 11.32607
[2,] 2.109451 4.851598 9.692904 12.59265
[3,] 2.735107 8.348120 6.679003 14.02135

$observed$corr
$observed$corr[[1]]
          [,1]        [,2]        [,3]       [,4]
[1,] 1.0000000  0.15930269  0.27719340  0.0773655
[2,] 0.1593027  1.00000000 -0.01981977 -0.3014780
[3,] 0.2771934 -0.01981977  1.00000000  0.1625010
[4,] 0.0773655 -0.30147800  0.16250101  1.0000000

$observed$corr[[2]]
           [,1]       [,2]       [,3]       [,4]
[1,]  1.0000000  0.1060559 -0.2681805  0.4033669
[2,]  0.1060559  1.0000000  0.2706491 -0.3458944
[3,] -0.2681805  0.2706491  1.0000000 -0.4174212
[4,]  0.4033669 -0.3458944 -0.4174212  1.0000000

$observed$corr[[3]]
            [,1]         [,2]         [,3]        [,4]
[1,]  1.00000000 -0.222455978  0.320940655 -0.09104018
[2,] -0.22245598  1.000000000 -0.002115642 -0.18530635
[3,]  0.32094065 -0.002115642  1.000000000  0.39979116
[4,] -0.09104018 -0.185306350  0.399791158  1.00000000
```

Again, the `hidden` element of `params` contains $\hat{\boldsymbol{p}}$, while the `observed` element contains the estimated observed variable distribution parameters. Here these consist of $\hat{\boldsymbol{\rho}} = (\hat{\boldsymbol{\rho}}_1, \ldots, \hat{\boldsymbol{\rho}}_K)$ in the `corr` element, and the $\hat{\sigma}$'s and $\hat{\delta}$'s in the `shape` and `decay` elements; that is, the elements of $\hat{\theta}_{y,d} = (\hat{\sigma}_{y,d}, \hat{\delta}_{y,d})$ are given by `shape[y,d]` and `decay[y,d]`.

2.2 Multiple-data models

The `exampleData3` set contains contains data generated from a layered model with $Z = 2$ in which the first observed data set comes from a univariate normal mixture and the second from a multivariate normal mixture. It contains the following elements:

X a list representing samples drawn from the distribution of the observed $\boldsymbol{\mathcal{X}}$, conditional on $\boldsymbol{\mathcal{Y}}$. It has elements `ed31`, a vector of length $N = 1000$, and `ed32`, an $N \times D_2$ matrix with $D_2 = 4$.

Y a list representing samples drawn from the distribution of the hidden $\boldsymbol{\mathcal{Y}}$ layer, conditional on $\mathcal{Y}_0$. It has elements `ed31` and `ed32`, both N-length vectors.

Y0 an N-length vector representing samples drawn from the distribution of the top-level hidden $\mathcal{Y}_0$.

Because both the normal distribution family and the layered topology are the defaults, a model with $K_0 = 2$, $K_1 = 2$, and $K_2 = 3$ may be fitted and printed as:

```
> mod3 <- mdmixmod(exampleData3$X, K = c(2, 3), K0 = 2)
> mod3

Layered (normal, normal) mixture model ('norm', 'mvnorm')
Data 'exampleData3$X' of size 1000-by-(1,4) fitted to 2 (2,3) components
Model statistics:
      iter       llik       qval        bic
    35.000  -9194.691 -12900.850 -18755.493
```

The `print` method for objects of class `mdmixmod` shows the distribution families used, the names of the actual distributions (here, univariate normal or `"norm"` for `X[[1]]` and multivariate normal or `"mvnorm"` for `X[[2]]`), data name and dimensions, the numbers of components $\boldsymbol{K} = (K_1, \ldots, K_Z)$ and K_0, and model statistics.

Like an object of class `mixmod`, an object of class `mdmixmod` is a list:

```
> names(mod3)

 [1] "N"                "Z"                "D"                "K"
 [5] "K0"               "X"                "npar"             "npar.hidden"
 [9] "npar.observed"    "iter"             "params"           "stats"
[13] "weights"          "pdfs"             "posterior"        "assignment"
[17] "iteration.params" "iteration.stats"  "topology"         "family"
[21] "distn"            "iter.max"         "dname"            "dattr"
```

Most of the elements have the same meaning as those in `mixmod`. However, the structure of `params` is somewhat more complicated, most easily visualized as a nested list using the `str` function of package `utils`:

```
> str(mod3$params)

List of 2
 $ hidden  :List of 2
  ..$ prob0: num [1:2] 0.433 0.567
  ..$ cprob:List of 2
  .. ..$ ed31: num [1:2, 1:2] 0.9099 0.1458 0.0901 0.8542
  .. ..$ ed32: num [1:2, 1:3] 0.4898 0.0756 0.3715 0.8224 0.1387 ...
 $ observed:List of 2
  ..$ ed31:List of 3
  .. ..$ mean: num [1:2] 3 -2.97
  .. ..$ var : num [1:2] 6.01 1.92
  .. ..$ sd  : num [1:2] 2.45 1.39
  ..$ ed32:List of 2
  .. ..$ mean: num [1:3, 1:4] 2.93 1.07 -1.99 1.95 -1.99 ...
  .. ..$ cov :List of 3
  .. .. ..$ : num [1:4, 1:4] 1.2279 -0.5309 0.2174 0.0205 -0.5309 ...
  .. .. ..$ : num [1:4, 1:4] 1.06049 -0.08953 -0.00389 -0.36244 -0.08953 ...
  .. .. ..$ : num [1:4, 1:4] 0.8525 0.1449 -0.0962 -0.0478 0.1449 ...
```

Here `hidden` has the elements `prob0` representing the estimated $\hat{\boldsymbol{p}}_0$ parameter of the distribution of $\mathcal{Y}_0$, and `cprob` (conditional probability) representing the estimated $\hat{\boldsymbol{Q}} = (\hat{\boldsymbol{Q}}_1, \ldots, \hat{\boldsymbol{Q}}_Z)$ parameters of the distributions of the $\mathcal{Y}_z$'s. Observe that the names of the elements of `cprob` are taken from the names of the elements of `X`. The `observed` element of `params` also has elements named after the elements of `X`, representing the estimated parameters of the observed variable distributions. Specifically, `ed31` has elements `mean` for $\hat{\boldsymbol{\mu}}_1 = (\hat{\mu}_{1:1}, \ldots, \hat{\mu}_{1:K_1})$, `var` for $\hat{\boldsymbol{\sigma}}_1^2 = (\hat{\sigma}_{1:1}^2, \ldots, \hat{\sigma}_{1:K_1}^2)$, and `sd` for $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_{1:1}, \ldots, \hat{\sigma}_{1:K_1})$, all vectors of length K_1; while `ed32` has elements `mean`, a $K_2 \times D_2$ matrix of which the k_2th row is $\hat{\boldsymbol{\mu}}_{k_2} = (\hat{\mu}_{k_2:1}, \ldots, \hat{\mu}_{k_2:D_2})$, and `cov`, a list of length K_2, of which the k_2th element is the $D_2 \times D_2$ positive definite matrix $\hat{\boldsymbol{\Sigma}}_{k_2}$.

The `posterior` element of an object of class `mdmixmod` is the $N \times K_0$ matrix of posterior probabilities for the top-level hidden component; that is, `posterior[n,k0]` represents the estimated $\Pr(\mathcal{Y}_0 = k_0)$ for the nth datum. For the posterior probabilities in the second-layer components, the `W` elements of the `weights` element of the object contains the weights used in the M-step of the EM algorithm for estimating the final set of parameters for the observed data portion of the model; `W[[z]][n,kz]` is the estimated $\Pr(\mathcal{Y}_z = k_z)$ for the nth datum.

The `exampleData4` set contains data generated from a chained model with $Z = 2$ in which the first observed data set comes from a univariate Weibull mixture and the second from a multivariate gamma mixture. It contains the following elements:

`X` a list representing samples drawn from the distribution of the observed $\boldsymbol{\mathcal{X}}$, conditional on $\boldsymbol{\mathcal{Y}}$. It has elements `ed41`, a vector of length $N = 1000$, and `ed42`, an $N \times D_2$ matrix with $D_2 = 4$.

`Y` a list representing samples drawn from the distribution of the hidden $\boldsymbol{\mathcal{Y}}$ layer, conditional on $\mathcal{Y}_0$ in the case of $\mathcal{Y}_1$, and on $\mathcal{Y}_1$ in the case of $\mathcal{Y}_2$. It has elements `ed41` and `ed42`, both N-length vectors.

Y0 an N-length vector representing samples drawn from the distribution of the top-level hidden $\mathcal{Y}_0$.

We fit and print a model with $K_0 = 2$, $K_1 = 2$, $K_2 = 3$, the chained topology, and the Weibull and gamma distribution families as follows:

```
> mod4 <- mdmixmod(exampleData4$X, K = c(2, 3), K0 = 2, topology = "chained",
+     family = c("wei", "gam"))
> mod4
```

```
Chained (Weibull, gamma) mixture model ('weisd', 'mvgamma')
Data 'exampleData4$X' of size 1000-by-(1,4) fitted to 2 (2,3) components
Model statistics:
     iter      llik      qval       bic
  121.000 -3451.138 -7527.333 -7268.387
```

The `posterior` and `weights` elements of the return value have the same meaning as in the layered model. The `params` element has some differences:

```
> str(mod4$params)
```

```
List of 2
 $ hidden  :List of 4
  ..$ prob0: num [1:2] 0.473 0.527
  ..$ probz:List of 2
  .. ..$ ed41: num [1:2] 0.448 0.552
  .. ..$ ed42: num [1:3] 0.335 0.447 0.218
  ..$ cprob:List of 2
  .. ..$ ed41: num [1:2, 1:2] 0.75 0.176 0.25 0.824
  .. ..$ ed42: num [1:2, 1:3] 0.566 0.148 0.157 0.682 0.277 ...
  ..$ rprob:List of 2
  .. ..$ ed41: num [1:2, 1:2] 0.793 0.214 0.207 0.786
  .. ..$ ed42: num [1:3, 1:2] 0.757 0.158 0.569 0.243 0.842 ...
 $ observed:List of 2
  ..$ ed41:List of 2
  .. ..$ shape: num [1:2] 0.995 2.008
  .. ..$ decay: num [1:2] 0.952 3.819
  ..$ ed42:List of 3
  .. ..$ shape: num [1:3, 1:4] 11.16 11.97 10.37 8.72 7.99 ...
  .. ..$ rate : num [1:3, 1:4] 0.918 2.151 2.777 3.81 4.873 ...
  .. ..$ corr :List of 3
  .. .. ..$ : num [1:4, 1:4] 1 -0.396 0.223 0.081 -0.396 ...
  .. .. ..$ : num [1:4, 1:4] 1 -0.1716 0.0392 -0.3387 -0.1716 ...
  .. .. ..$ : num [1:4, 1:4] 1 0.1425 0.1383 -0.0242 0.1425 ...
```

As in the layered model, the `prob0` and `cprob` elements of `hidden` represent the estimated $\hat{\boldsymbol{p}}_0$ and $\hat{\boldsymbol{Q}}$ parameters respectively (although recall that $\boldsymbol{Q}$ has a somewhat different meaning in the layered model than in the chained.) The `probz` element represents the estimated marginal probabilities for the $\mathcal{Y}_z$'s, that is, $\hat{p}_{z:y_z} = \widehat{\Pr}(\mathcal{Y}_z = y_z)$, and `rprob` represents the estimated "reverse conditional probabilities," that is, $\hat{r}_{z:y_z,y_{z-1}} = \widehat{\Pr}(\mathcal{Y}_{z-1} = y_{z-1} \mid \mathcal{Y}_z = y_z)$. Both `probz` and `rprob` are functions of `prob0` and `cprob`, and are reported only for convenience. In `observed`, element `ed41` has elements `shape` for $\hat{\boldsymbol{\sigma}}_1 = (\hat{\sigma}_{1:1}, \ldots, \hat{\sigma}_{1:K_1})$ and `decay` for elements $\hat{\boldsymbol{\delta}}_1 = (\hat{\delta}_{1:1}, \ldots, \hat{\delta}_{1:K_1})$. Element `ed42` has elements `shape`, a $K_2 \times D$ matrix of which the (k_2, d)th element is $\hat{\sigma}_{2:k_2,d}$, the estimated shape parameter for the marginal distribution of $\mathcal{X}_{2:d}$ (the random variable from which the dth column of `X[[2]]` is drawn) given that $\mathcal{Y}_2 = k_2$; `rate`, a $K_2 \times D$ matrix of which the (k_2, d)th element is $\hat{\lambda}_{2:k_2,d}$, the estimated rate parameter for the marginal distribution of $\mathcal{X}_{2:d}$ given that $\mathcal{Y}_2 = k_2$; and `corr`, a K_2-length list of which the k_2th element is $\hat{\boldsymbol{\rho}}_{2:k_2}$, the estimated copula correlation matrix for the distribution of $\boldsymbol{\mathcal{X}}_2$ given that $\mathcal{Y}_2 = k_2$.

3 Evaluating model fit

Evaluations of model fit may be broadly divided into "internal" and "external" evaluations; in the first case, we wish to know how well the model fits the data; in the second, we wish to know how well the model performs its function, that is, classifying the data into particular groups. For internal evaluation, the primary tool is the log-likelihood $\mathcal{L}$ and functions of $\mathcal{L}$ such as the BIC (Schwarz, 1978) defined here as

$$\text{BIC} = 2\mathcal{L} - |\theta| \log N \tag{15}$$

where $|\Theta|$ is the size of the parameter space. In general, given two competing models, the model with the higher BIC is preferred. Log-likelihood and BIC are reported when an object of class `mixmod` or `mdmixmod` is printed, as seen in §2. They can also be extracted using the `logLik` and `bic` functions, as seen in the following example.

```
> logLik(mod1)

[1] -1771.990

> bic(mod1)

[1] -3578.518

> mods.ed1 <- lapply(1:4, function(K) mixmod(exampleData1$X, K))
> cbind(llik = sapply(mods.ed1, logLik), bic = sapply(mods.ed1,
+     bic))

          llik       bic
[1,] -2163.700 -4341.215
[2,] -1771.990 -3578.518
[3,] -1771.955 -3599.172
[4,] -1771.946 -3619.878
```

Log-likelihood cannot be used directly to choose the number of components, because it always increases with the number of model parameters. But BIC peaks here at $K = 2$, indicating that the two-component model is a good fit for the observed data in `exampleData1`.

BIC can also be used to choose between two different distributions for modeling data, so long as those distributions have the same support, such as $[0, \infty)$ for the Weibull and gamma:

```
> mod2gam <- mixmod(exampleData2$X, 3, family = "gam")
> bic(mod2)

[1] 5506.626

> bic(mod2gam)

[1] 5359.246
```

We see here that the Weibull model outperforms the gamma model on the observed data in `exampleData2`.

The expected log-likelihood for the complete data, that is, the value of the Q-function in the final iteration of the EM algorithm, can be extracted with the `qval` function. The value of this function is the sum of hidden and observed log-likelihood terms, as reported by the `qfun` function:

```
> qval(mod3)

[1] -12900.88

> qfun(mod3)

   hidden  observed
-5074.843 -7826.038
```

Log-likelihood should converge smoothly during execution of the EM algorithm. To examine the convergence characteristics, we can plot the iteration history using the `convergence.plot` function:

```
> convergence.plot(mod4)
```

The resulting plot is shown in Figure 3.

External evaluation is possible only when at least some of the true component labels are known. In the case of the `exampleData` sets, where all the components are known, the ROC (receiver operating characteristic) curve is a useful tool for evaluation. In `lcmix` as long as the suggested `ROCR` package is installed, one or more ROC curves may be drawn on the same plot using the `multiROC` function and its associated `plot` method:

```
> plot(multiROC(list(mod1 = mod1$posterior[, 1]), labels = (exampleData1$Y ==
+     1)))
```

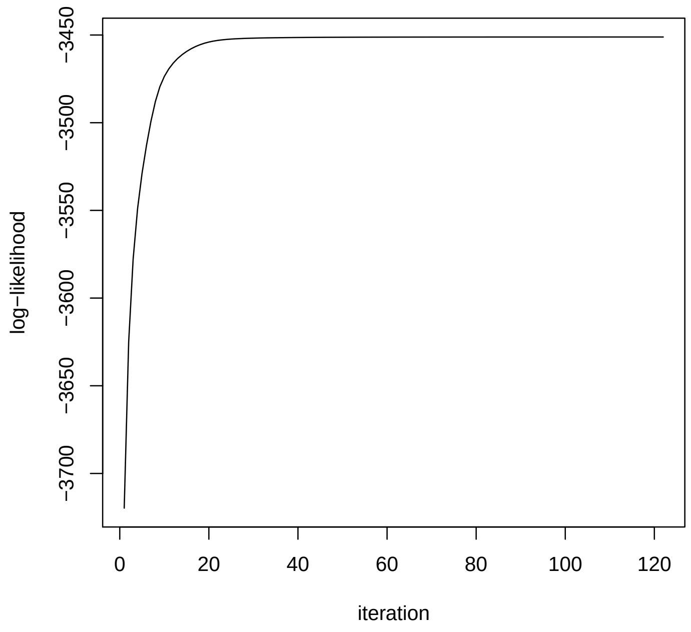

Figure 3: Convergence plot for `mod4` fitted to the observed data in `exampleData4`.

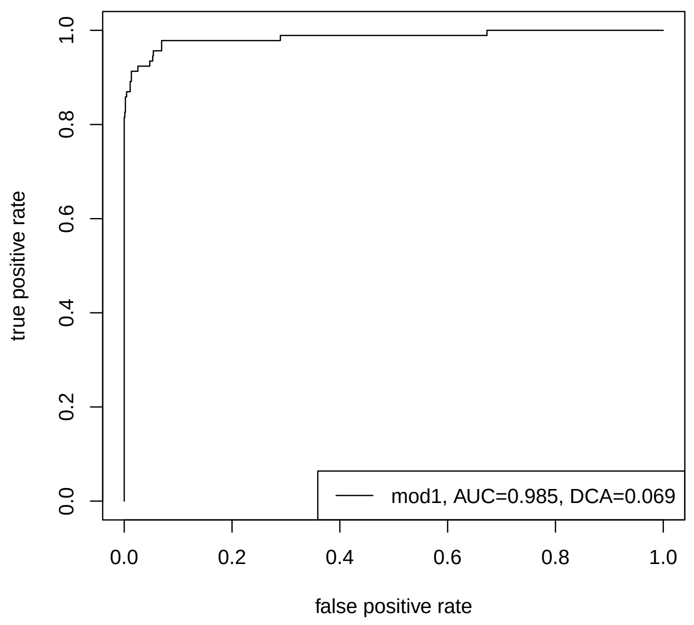

Figure 4: ROC plot for `mod1`'s predictions of membership in the first component in `exampleData1`.

The resulting plot, shown in Figure 4, allows us to evaluate how well the model predicts membership in the first component. A large AUC, or area under the ROC curve, close to 1, and a small DCA, or distance of closest approach of the curve to the point (0, 1), are both indicators of a good predictive model; the closer AUC is to 1 and DCA is to 0, the better the prediction.

In the event that only some of the labels are known — that is, some of the data are known to belong to a particular category, but we know or suspect that other, unlabeled data are also members of the category — a quasi-ROC plot in which the x-axis of the plot represents all positives rather than false positives, may be used. This is reasonable when the overall rate of membership in the category of interest is thought to be small. As an example, suppose we have incomplete knowledge about membership in the first component in `exampleData2`:

```
> set.seed(123)
> N <- length(exampleData2$Y)
> labels2 <- (exampleData2$Y == 1) & as.logical(rbinom(N, 1, 0.5))
> plot(multiROC(list(mod2 = mod2$posterior[, 1]), labels2, quasi = TRUE))
```

The resulting plot is shown in Figure 5.

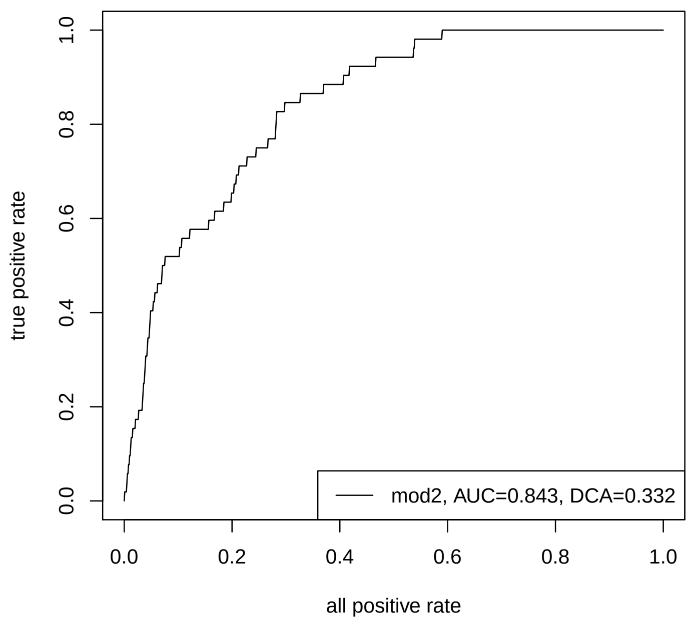

Figure 5: Quasi-ROC plot for `mod2`'s predictions of membership in the first component in `exampleData2`, with partial knowledge.

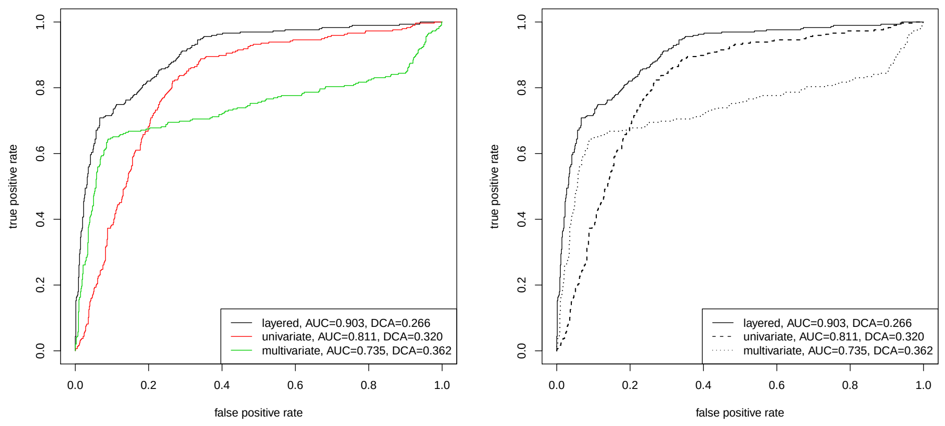

Figure 6: Multiple ROC plots for multiple-data and single-data predictions of membership in the first top-level component in `exampleData3`.

As its name implies, `multiROC` also allows us to draw multiple ROC curves on a single plot. For example, we may compare the effectiveness of the layered model at predicting top-level (rvY_0) first-component membership to that of the individual data sources in `exampleData3`, in both color and black-and-white:

```
> mod31 <- mixmod(exampleData3$X$ed31, 2)
> mod32 <- mixmod(exampleData3$X$ed32, 3)
> preds3 <- list(layered = mod3$posterior[, 1], univariate = mod31$posterior[,
+     1], multivariate = mod32$posterior[, 1])
> labels3 <- (exampleData3$Y0 == 1)
> par(mfrow = c(1, 2))
> plot(multiROC(preds3, labels3))
> plot.multiROC.bw(multiROC(preds3, labels3))
> par(mfrow = c(1, 1))
```

The resulting plot is shown in Figure 6.

Finally, `multiROC` can plot multiple predictors against multiple labels. For this example, we compare the chained model's top-level first-component membership predictions to predictions of membership in the first component at the $\mathcal{Y}_z$ level:

```
> preds4 <- list()
> preds4$chained <- mod4$posterior[, 1]
> preds4$univariate <- mod4$weights$W$ed41[, 1]
> preds4$multivariate <- mod4$weights$W$ed42[, 1]
> labels4 <- list()
> labels4$chained <- (exampleData4$Y0 == 1)
> labels4$univariate <- (exampleData4$Y$ed41 == 1)
> labels4$multivariate <- (exampleData4$Y$ed42 == 1)
```

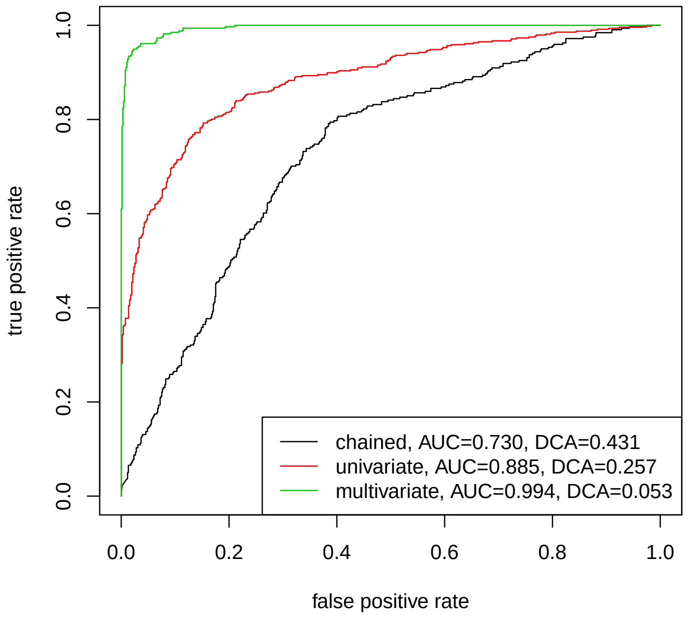

Figure 7: Multiple ROC plots for chained model predictions in `exampleData4`.

```
> mr4 <- multiROC(preds4, labels4)
> plot(mr4)
```

The resulting plot is shown in Figure 7. The performance measures are present in the `meas` element of a `multiROC` object:

```
> mr4$meas

$chained
      auc       dca
0.7298942 0.4307227

$univariate
      auc       dca
0.8850583 0.2571570

$multivariate
       auc        dca
0.99414175 0.05294699
```

4 Simulating data

The key functions for simulating data are `simulate.mixdata` for simulation of a single data source, `simulate.mdmixdata` for simulation of multiple data sources, and `simulate.from.fit` for simulation from an already fitted (single-data or multiple-data) model. All simulation functions return both the observed and hidden data. The use of the first two functions to create the `exampleData` sets included in `lcmix`, is illustrated below, followed by the used of the third function to create a data set similar to `exampleData1`

To begin, we set a random seed to ensure reproducibility, and establish the size of the data we wish to simulate:

```
> set.seed(123)
> N <- 1000
```

Next we set up the parameters for a very simple single-data model using a univariate normal mixture, and call `simulate.mixdata` to create the data sets `exampleData1`:

```
> p <- c(0.1, 0.9)
> mu <- c(3, -3)
> sigma <- c(3, 1)
> params1 <- list(hidden = list(prob = p), observed = list(mean = mu,
+     sd = sigma, var = sigma^2))
> exampleData1 <- simulate.mixdata(N, distn = "norm", params = params1)
```

That is, we have a univariate normal mixture with two components ($K = 2$) with hidden variable distribution parameters $p_1 = 0.1, p_2 = 0.9$, and observed variable distribution parameters $\mu_1 = 3,\ \mu_2 = -3$ and $\sigma_1 = 3,\ \sigma_2 = 1$. The structure of the `params` argument to `simulate.mixdata` is important; it reflects the structure of the parameters returned by the model-fitting functions discussed in §2. The `var` element in the `observed` list in `params1` is not strictly necessary, as the normal distribution simulation requires only the standard deviation, but is included to match the full structure.

The `exampleData1` data set is a list with the observed data in element `X` and the hidden data in the element `Y`, both vectors of length `N`:

```
> names(exampleData1)

[1] "X" "Y"

> dim(as.data.frame(exampleData1))

[1] 1000    2

> as.data.frame(exampleData1)[1:10, ]
```

```
           X Y
1  -2.159460 2
2  -3.285845 2
3  -2.495874 2
4  -4.155917 2
5   1.194321 1
6  -3.127149 2
7  -4.941518 2
8  -1.818819 2
9  -1.140089 2
10 -1.925988 2
```

Note that as expected, 1 out of 10 of the simulated data are assigned to component 1.

For a slightly more complicated example, we next simulate data from a multivariate Weibull (WeiSD) mixture distribution with $K = 3$ components and observed data width $D = 4$:

```
> p <- c(0.1, 0.7, 0.2)
> s <- matrix(12:1, ncol = 4)
> d <- matrix(1:12, ncol = 4)
> rho <- lapply(1:3, function(k) cor(matrix(rnorm(40), ncol = 4)))
> params2 <- list(hidden = list(prob = p), observed = list(shape = s,
+     decay = d, corr = rho))
> exampleData2 <- simulate.mixdata(N, "mvweisd", params2)
> names(exampleData2)

[1] "X" "Y"

> class(exampleData2$X)

[1] "matrix"

> dim(exampleData2$X)

[1] 1000    4

> class(exampleData2$Y)

[1] "integer"

> length(exampleData2$Y)

[1] 1000

> rho
```

```
[[1]]
           [,1]        [,2]        [,3]        [,4]
[1,] 1.00000000  0.03986286  0.42284624  0.01966361
[2,] 0.03986286  1.00000000 -0.34271191 -0.29535982
[3,] 0.42284624 -0.34271191  1.00000000  0.09747647
[4,] 0.01966361 -0.29535982  0.09747647  1.00000000

[[2]]
           [,1]       [,2]       [,3]       [,4]
[1,]  1.0000000  0.1289651 -0.2218877  0.3319994
[2,]  0.1289651  1.0000000  0.2156014 -0.3809145
[3,] -0.2218877  0.2156014  1.0000000 -0.3508474
[4,]  0.3319994 -0.3809145 -0.3508474  1.0000000

[[3]]
            [,1]        [,2]       [,3]        [,4]
[1,]  1.00000000 -0.21040035 0.34888749  0.04319041
[2,] -0.21040035  1.00000000 0.02681911 -0.20586668
[3,]  0.34888749  0.02681911 1.00000000  0.37659758
[4,]  0.04319041 -0.20586668 0.37659758  1.00000000
```

Note that the shape and decay parameters are $K \times D$ matrices, such that `s[y,d]` and `d[y,d]` are the shape and decay used to generate values in `X[,d]` where `Y == d`. The correlation parameter `rho` is a list such that the columns of `X` are simulated using copula correlation `rho[[d]]` where `Y == d`.

The `exampleData3` set is generated using `simulate.mdmixmod` and parameters having the same structure as the `params` element of an object of class `mdmixmod`; see §2.2 for details.

```
> p0 <- c(0.3, 0.7)
> Q1 <- matrix(c(0.9, 0.1, 0.3, 0.7), nrow = 2, byrow = TRUE)
> Q2 <- matrix(c(0.6, 0.2, 0.2, 0.1, 0.8, 0.1), nrow = 2, byrow = TRUE)
> Q <- list(ed31 = Q1, ed32 = Q2)
> mu1 <- c(3, -3)
> mu2 <- matrix(c(3, 2, 1, 4, 1, -2, 0, -1, -2, -4, -3, -5), nrow = 3,
+     byrow = TRUE)
> colnames(mu2) <- paste("ed32", 1:4, sep = "")
> sigmasq1 <- 1 + rexp(2)
> sigma1 <- sqrt(sigmasq1)
> Sigma2 <- lapply(1:3, function(k) {
+     Z = matrix(rnorm(100), ncol = 4)
+     colnames(Z) = paste("ed32", 1:4, sep = "")
+     mlecov(Z)
+ })
> theta.ed31 <- list(mean = mu1, var = sigmasq1, sd = sigma1)
> theta.ed32 <- list(mean = mu2, cov = Sigma2)
> params3 <- list(hidden = list(prob0 = p0, cprob = Q), observed = list(ed31 = theta.ed31,
+     ed32 = theta.ed32))
> exampleData3 <- simulate.mdmixdata(N, c("norm", "mvnorm"), params3)
```

The `exampleData4` set follows the same pattern; note that the `rprob` and `probz` elements of the `hidden` parameter element are not needed as input, although they may be included without effect.

```
> shape1 <- c(1, 2)
> decay1 <- c(1, 4)
> theta.ed41 <- list(shape = shape1, decay = decay1)
> shape2 <- s
> rate2 <- d
> corr2 <- lapply(Sigma2, function(Sigma) {
+     res = cov2cor(Sigma)
+     colnames(res) = rownames(res) = paste("ed42", 1:4, sep = "")
+     return(res)
+ })
> theta.ed42 <- list(shape = shape2, rate = rate2, corr = corr2)
> names(Q) <- c("ed41", "ed42")
> params4 <- list(hidden = list(prob0 = p0, cprob = Q), observed = list(ed41 = theta.ed41,
+     ed42 = theta.ed42))
> exampleData4 <- simulate.mdmixdata(N, c("weisd", "mvgamma"),
+     params4, topology = "chained")
```

Given a fitted model, we may simply simulate from the parameters of the model using the `simulate.from.fit` convenience function:

```
> exampleData1sim <- simulate.from.fit(mod1)
> mod1sim <- mixmod(exampleData1sim$X, 2)
> mod1$params

$hidden
$hidden$prob
[1] 0.09250383 0.90749617



$observed
$observed$mean
[1]  2.823488 -2.983441

$observed$var
[1] 7.329986 1.020609

$observed$sd
[1] 2.707395 1.010252

> mod1sim$params
```

```
$hidden
$hidden$prob
[1] 0.0888815 0.9111185


$observed
$observed$mean
[1]  3.160794 -3.006016

$observed$var
[1] 7.916763 1.045567

$observed$sd
[1] 2.813674 1.022530
```

The first line in the above example is equivalent to

```
> exampleData1sim <- simulate.mixdata(mod1$N, mod1$distn, mod1$params)
```

but is somewhat simpler; `simulate.from.fit` is particularly convenient when dealing with objects of class `mdmixdata`.

References

T. Benaglia, D. Chauveau, D.R. Hunter, and D. Young. `mixtools`: An R package for analyzing finite mixture models. *Journal of Statistical Software*, 32(6):1–29, 2009. URL `http://www.jstatsoft.org/v32/i06/`.

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

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

G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* Wiley, second edition, 2008.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6 (2):461–464, 1978.

P.X. Song. Multivariate dispersion models generated from Gaussian copula. *Scandinavian Journal of Statistics*, 27(2):305–320, 2000. URL `http://www.jstor.org/stable/4616605`.