Complement: Finite Mixture Model Diagnostics Using Resampling Methods

Bettina Grün
Johannes Kepler Universität Linz

Friedrich Leisch
Universität für Bodenkultur Wien

Abstract

This paper illustrates the implementation of resampling methods in **flexmix** as well as the application of resampling methods for model diagnostics of fitted finite mixture models. Convenience functions to perform these methods are available in package **flexmix**. The results of the application to an artificial example and the `seizure` data set as described in Grün and Leisch (2010) are reproduced.

Keywords:˜R, finite mixture models, resampling, bootstrap.

1. Implementation of resampling methods

The proposed framework for model diagnostics using resampling equally allows to investigate model fit for all kinds of mixture models. The procedure is applicable to mixture models with different component specific models and does not impose any limitation such as for example on the dimension of the parameter space of the component specific model. In addition to the fitting step different component specific models only require different random number generators for the parametric bootstrap.

The `boot()` function in **flexmix** is a generic S4 function with a method for fitted finite mixtures of class `"flexmix"` and is applicable to general finite mixture models. The function with arguments and their defaults is given by:

```
boot(object, R, sim = c("ordinary", "empirical", "parametric"),
  initialize_solution = FALSE, keep_weights = FALSE, keep_groups = TRUE,
  verbose = 0, control, k, model = FALSE, ...)
```

The interface is similar to the `boot()` function in package **boot** (Davison and Hinkley 1997; Canty and Ripley 2010). The `object` is a fitted finite mixture of class `"flexmix"` and `R` denotes the number of resamples. The possible bootstrapping method are `"empirical"` (also available as `"ordinary"`) and `"parametric"`. For the parametric bootstrap sampling from the fitted mixture is performed using `rflexmix()`. For mixture models with different component specific models `rflexmix()` requires a sampling method for the component specific model. Argument `initialize_solution` allows to select if the EM algorithm is started in the original finite mixture solution or if random initialization is performed. The fitted mixture model might contain weights and group indicators. The weights are case weights and allow to reduce the amount of data if observations are identical. This is useful for example for latent class

analysis of multivariate binary data. The argument `keep_weights` allows to indicate if they should be kept for the bootstrapping. Group indicators allow to specify that the component membership is identical over several observations, e.g., for repeated measurements of the same individual. Argument `keep_groups` allows to indicate if the grouping information should also be used in the bootstrapping. `verbose` indicates if information on the progress should be printed. The `control` argument allows to control the EM algorithm for fitting the model to each of the bootstrap samples. By default the `control` argument is extracted from the fitted model provided by `object`. `k` allows to specify the number of components and by default this is also taken from the fitted model provided. The `model` argument determines if also the model and the weights slot for each sample are stored and returned. The returned object is of class `"FLXboot"` and otherwise only contains the fitted parameters, the fitted priors, the log likelihoods, the number of components of the fitted mixtures and the information if the EM algorithm has converged.

The likelihood ratio test is implemented based on `boot()` in function `LR_test()` and returns an object of class `"htest"` containing the number of valid bootstrap replicates, the p-value, the double negative log likelihood ratio test statistics for the original data and the bootstrap replicates. The `plot` method for `"FLXboot"` objects returns a parallel coordinate plot with the fitted parameters separately for each of the components.

2. Artificial data set

In the following a finite mixture model is used as the underlying data generating process which is theoretically not identifiable. We are assuming a finite mixture of linear regression models with two components of equal size where the coverage condition is not fulfilled (Hennig 2000). Hence, intra-component label switching (Grün and Leisch 2010) is possible, i.e., there exist two parameterizations implying the same mixture distribution which differ how the components between the covariate points are combined.

We assume that one measurement per object and a single categorical regressor with two levels are given. The usual design matrix for a model with intercept uses the two covariate points $\mathbf{x}_1 = (1, 0)'$ and $\mathbf{x}_2 = (1, 1)'$. The mixture distribution is given by

$$H(y|\mathbf{x}, \Theta) \quad = \quad \frac{1}{2}N(\mu_1, 0.1) + \frac{1}{2}N(\mu_2, 0.1),$$

where $\mu_k(\mathbf{x}) = \mathbf{x}'\boldsymbol{\alpha}_k$ and $N(\mu, \sigma^2)$ is the normal distribution.

Now let $\mu_1(\mathbf{x}_1) = 1$, $\mu_2(\mathbf{x}_1) = 2$, $\mu_1(\mathbf{x}_2) = -1$ and $\mu_2(\mathbf{x}_2) = 4$. As Gaussian mixture distributions are generically identifiable the means, variances and component weights are uniquely determined in each covariate point given the mixture distribution. However, as the coverage condition is not fulfilled, the two possible solutions for $\boldsymbol{\alpha}$ are:

Solution 1: $\boldsymbol{\alpha}_1^{(1)} = (2, \quad 2)'$, $\boldsymbol{\alpha}_2^{(1)} = (1, -2)'$,

Solution 2: $\boldsymbol{\alpha}_1^{(2)} = (2, -3)'$, $\boldsymbol{\alpha}_2^{(2)} = (1, \quad 3)'$.

We specify this artificial mixture distribution using `FLXdist()`. `FLXdist()` returns an unfitted finite mixture of class `"FLXdist"`. The class of fitted finite mixture models `"flexmix"` extends class `"FLXdist"`. Each component follows a normal distribution. The parameters specified

in a named list therefore consist of the regression coefficients and the standard deviation. Function `FLXdist()` has an argument `formula` for specifying the regression in each of the components, an argument `k` for the component weights and `components` for the parameters of each of the components.

```
> library("flexmix")
> Component_1 <- list(Model_1 = list(coef = c(1, -2), sigma = sqrt(0.1)))
> Component_2 <- list(Model_1 = list(coef = c(2,  2), sigma = sqrt(0.1)))
> ArtEx.mix <- FLXdist(y ~ x, k = rep(0.5, 2),
+                      components = list(Component_1, Component_2))
```

We draw a balanced sample with 50 observations in each covariate point from the mixture model using `rflexmix()` after defining the data points for the covariates. `rflexmix()` can either have an unfitted or a fitted finite mixture as input. For unfitted mixtures data has to be provided using the `newdata` argument. For already fitted mixtures data can be optionally provided, otherwise the data used for fitting the mixture is used.

```
> ArtEx.data <- data.frame(x = rep(0:1, each = 100/2))
> set.seed(123)
> ArtEx.sim <- rflexmix(ArtEx.mix, newdata = ArtEx.data)
> ArtEx.data$y <- ArtEx.sim$y[[1]]
> ArtEx.data$class <- ArtEx.sim$class
```

In Figure˜1 the sample is plotted together with the two solutions for combining x_1 and x_2, i.e., this illustrates intra-component label switching.

We fit a finite mixture to the sample using `stepFlexmix()`.

```
> set.seed(123)
> ArtEx.fit <- stepFlexmix(y ~ x, data = ArtEx.data, k = 2, nrep = 5,
+                          control = list(iter = 1000, tol = 1e-8, verbose = 0))

2 : * * * * *
```

The fitted mixture can be inspected using `summary()` and `parameters()`.

```
> summary(ArtEx.fit)

Call:
stepFlexmix(y ~ x, data = ArtEx.data, control = list(iter = 1000,
    tol = 1e-08, verbose = 0), k = 2, nrep = 5)

       prior size post>0 ratio
Comp.1  0.56   55     77 0.714
Comp.2  0.44   45     65 0.692

'log Lik.' -82.52413 (df=7)
AIC: 179.0483   BIC: 197.2845
```

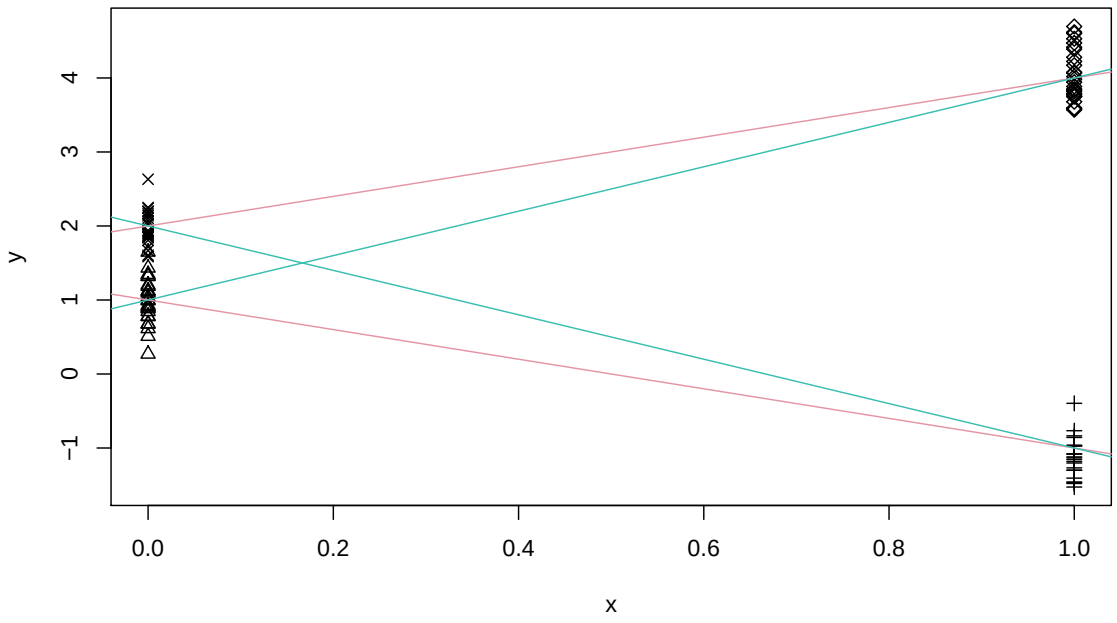

Figure 1: Balanced sample from the artificial example with the two theoretical solutions.

```
> parameters(ArtEx.fit)

                    Comp.1     Comp.2
coef.(Intercept) 1.9130903  0.9269647
coef.x           2.1270909 -2.0597489
sigma            0.3167696  0.2738558
```

Obviously the fitted mixture parameters correspond to the parameterization we used to specify the mixture distribution. Using standard asymptotic theory to analyze the fitted mixture model gives the following estimates for the standard deviations.

```
> ArtEx.refit <- refit(ArtEx.fit)
> summary(ArtEx.refit)

$Comp.1
            Estimate Std. Error z value  Pr(>|z|)
(Intercept) 1.914496   0.072060  26.568 < 2.2e-16 ***
x           2.125685   0.093203  22.807 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

$Comp.2
             Estimate Std. Error z value  Pr(>|z|)
(Intercept)  0.927075   0.068756  13.484 < 2.2e-16 ***
```

```
x           -2.059859   0.089780 -22.943 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fitted mixture can also be analyzed using resampling techniques. For analyzing the stability of the parameter estimates where the possibility of identifiability problems is also taken into account the parametric bootstrap is used with random initialization. Function `boot()` can be used for empirical or parametric bootstrap (specified by the argument `sim`). The logical argument `initialize_solution` specifies if the initialization is in the original solution or random. By default random initialization is made. The number of bootstrap samples is set by the argument `R`. Please note that the arguments are chosen to correspond to those for function `boot` in package **boot** (Davison and Hinkley 1997).

Only a few number of bootstrap samples are drawn to keep the amount of time needed to run the vignette within reasonable limits. However, for a sensible application of the bootstrap methods at least `R` equal to 100 should be used. If the output for this setting has been saved, it is loaded and used in the further analysis. Please see the appendix for the code for generating the saved R output.

```
> set.seed(123)
> ArtEx.bs <- boot(ArtEx.fit, R = 15, sim = "parametric")
> if ("boot-output.rda" %in% list.files()) load("boot-output.rda")
> ArtEx.bs

Call:
boot(ArtEx.fit, R = 15, sim = "parametric")
```

Function `boot()` returns an object of class `"FLXboot"`. The default plot compares the bootstrap parameter estimates to the confidence intervals derived using standard asymptotic theory in a parallel coordinate plot (see Figure˜2). Clearly two groups of parameter estimates can be distinguished which are about of equal size. One subset of the parameter estimates stays within the confidence intervals induced by standard asymptotic theory, while the second group corresponds to the second solution and clusters around these parameter values.

In the following the DIP-test is applied to check if the parameter estimates follow a unimodal distribution. This is done for the aggregated parameter esimates where unimodality implies that this parameter is not suitable for imposing an ordering constraint which induces a unique labelling. For the separate component analysis which is made after imposing an ordering constraint on the coefficient of x rejection the null hypothesis of unimodality implies that identifiability problems are present, e.g.˜due to intra-component label switching.

```
> require("diptest")
> parameters <- parameters(ArtEx.bs)
> Ordering <- factor(as.vector(apply(matrix(parameters[,"coef.x"],
+                                           nrow = 2), 2, order)))
> Comp1 <- parameters[Ordering == 1,]
> Comp2 <- parameters[Ordering == 2,]
> dip.values.art <- matrix(nrow = ncol(parameters), ncol = 3,
```

```
> print(plot(ArtEx.bs, ordering = "coef.x", col = Colors))
```

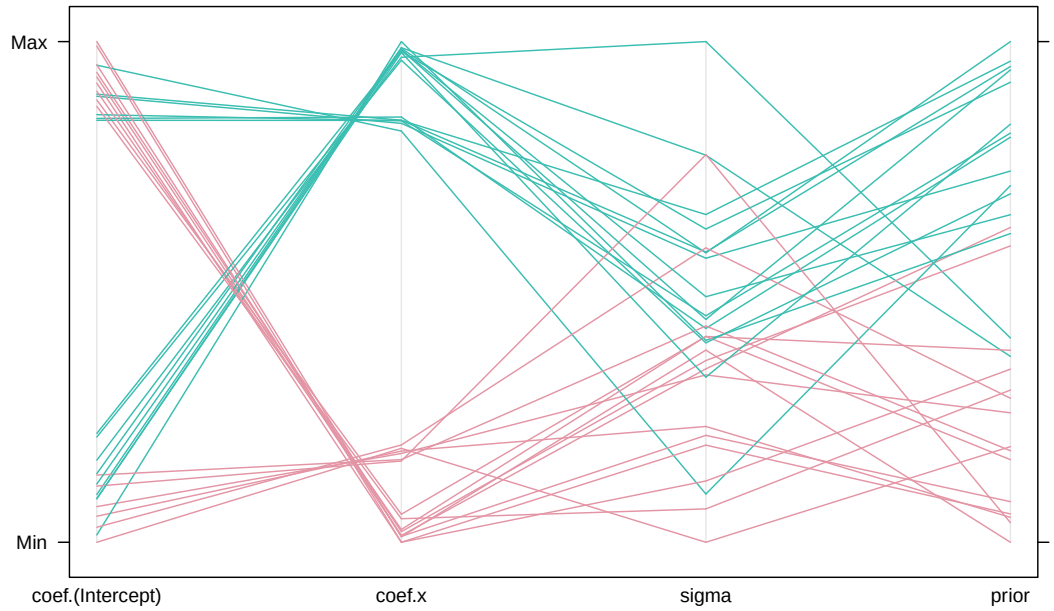

Figure 2: Diagnostic plot of the bootstrap results for the artificial example.

```
+                           dimnames=list(colnames(parameters),
+                             c("Aggregated", "Comp 1", "Comp 2")))
> dip.values.art[,"Aggregated"] <- apply(parameters, 2, dip)
> dip.values.art[,"Comp 1"] <- apply(Comp1, 2, dip)
> dip.values.art[,"Comp 2"] <- apply(Comp2, 2, dip)
> dip.values.art
```

```
                  Aggregated     Comp 1     Comp 2
coef.(Intercept) 0.18549534 0.16915873 0.17007200
coef.x           0.19083739 0.15364192 0.16044336
sigma            0.05397328 0.07343813 0.08717279
prior            0.07764922 0.07628739 0.07628739
```

The critical value for column `Aggregated` is 0.088 and for the columns of the separate components 0.119. The component sizes as well as the standard deviations follow a unimodal distribution for the aggregated data as well as for each of the components. The regression coefficients are multimodal for the aggregate data as well as for each of the components. While from the aggregated case it might be concluded that imposing an ordering constraint on the intercept or the coefficient of x is suitable, the component-specific analyses reveal that a unique labelling was not achieved.

3. Seizure

In Wang, Puterman, Cockburn, and Le (1996) a Poisson mixture regression is fitted to data from a clinical trial where the effect of intravenous gammaglobulin on suppression of epileptic seizures is investigated. The data used were 140 observations from one treated patient, where treatment started on the 28$^{\text{th}}$ day. In the regression model three independent variables were included: treatment, trend and interaction treatment-trend. Treatment is a dummy variable indicating if the treatment period has already started. Furthermore, the number of parental observation hours per day were available and it is assumed that the number of epileptic seizures per observation hour follows a Poisson mixture distribution. The number of epileptic seizures per parental observation hour for each day are plotted in Figure˜3. The fitted mixture distribution consists of two components which can be interpreted as representing 'good' and 'bad' days of the patients.

The mixture model can be formulated by

$$H(y|\mathbf{x}, \Theta) = \pi_1 P(\lambda_1) + \pi_2 P(\lambda_2),$$

where $\lambda_k = e^{\mathbf{x}'\boldsymbol{\alpha}_k}$ for $k = 1, 2$ and $P(\lambda)$ is the Poisson distribution.

The data is loaded and the mixture fitted with two components.

```
> data("seizure", package = "flexmix")
> model <- FLXMRglm(family = "poisson", offset = log(seizure$Hours))
> control <- list(iter = 1000, tol = 1e-10, verbose = 0)
> set.seed(123)
> seizMix <- stepFlexmix(Seizures ~ Treatment*log(Day),
+                            data = seizure, k = 2, nrep = 5,
+                            model = model, control = control)

2 : * * * * *
```

The fitted regression lines for each of the two components are shown in Figure˜3.

The parameteric bootstrap with random initialization is used to investigate identifiability problems and parameter stability. The diagnostic plot is given in Figure˜3. The coloring is according to an ordering constraint on the intercept. Clearly the parameter estimates corresponding to the solution where the bad days from the base period are combined with the good days from the treatement period and vice versa for the good days of the base period can be distinguished and indicate the slight identifiability problems of the fitted mixture.

```
> parameters <- parameters(seizMix.bs)
> Ordering <- factor(as.vector(apply(matrix(parameters[,"coef.(Intercept)"],
+                                           nrow = 2), 2, order)))
> Comp1 <- parameters[Ordering == 1,]
> Comp2 <- parameters[Ordering == 2,]
```

For applying the DIP test also an ordering constraint on the intercept is used. The critical value for column `Aggregated` is 0.088 and for the columns of the separate components 0.119.

```
> par(mar = c(5, 4, 2, 0) + 0.1)
> plot(Seizures/Hours~Day, data=seizure, pch = as.integer(seizure$Treatment))
> abline(v=27.5, lty=2, col="grey")
> matplot(seizure$Day, fitted(seizMix)/seizure$Hours, type="l",
+         add=TRUE, col=1, lty = 1, lwd = 2)
```

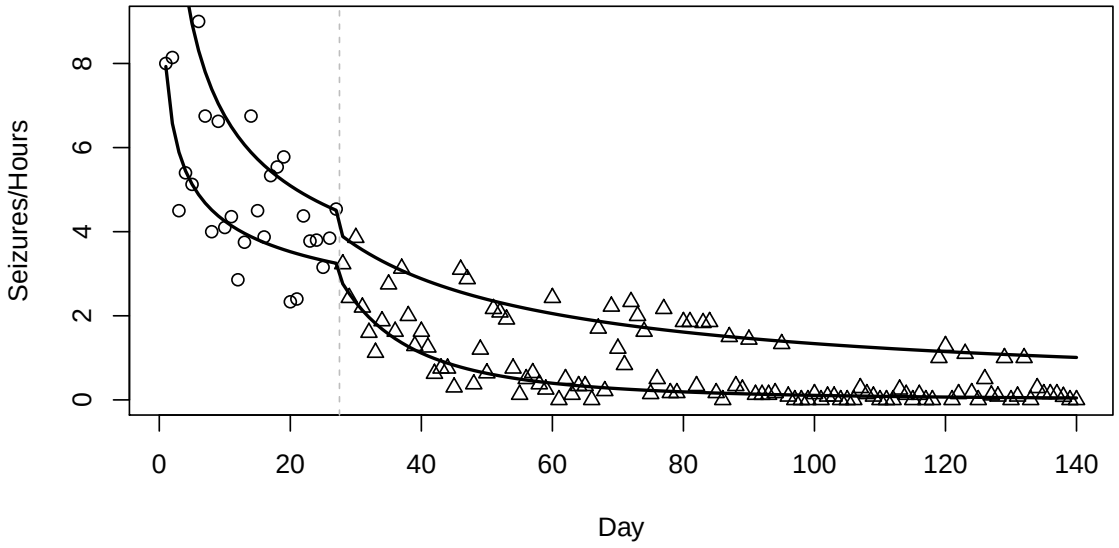

Figure 3: Seizure data with the fitted values for the Wang *et˜al.* model. The plotting character for the observed values in the base period is a circle and for those in the treatment period a triangle.

```
> dip.values.art <- matrix(nrow = ncol(parameters), ncol = 3,
+                          dimnames=list(colnames(parameters),
+                            c("Aggregated", "Comp 1", "Comp 2")))
> dip.values.art[,"Aggregated"] <- apply(parameters, 2, dip)
> dip.values.art[,"Comp 1"] <- apply(Comp1, 2, dip)
> dip.values.art[,"Comp 2"] <- apply(Comp2, 2, dip)
> dip.values.art

                           Aggregated     Comp 1     Comp 2
coef.(Intercept)           0.11424627 0.08251065 0.06239055
coef.TreatmentYes          0.14724776 0.11290550 0.08763688
coef.log(Day)              0.08771107 0.07954399 0.04785066
coef.TreatmentYes:log(Day) 0.16430740 0.08548047 0.08940126
prior                      0.15660112 0.08275558 0.08275558
```

For the aggregate results the hypothesis of unimodality cannot be rejected for the trend. For the component-specific analyses unimodality cannot be rejected only for the intercept

```
> set.seed(123)
> seizMix.bs <- boot(seizMix, R = 15, sim = "parametric")
> if ("boot-output.rda" %in% list.files()) load("boot-output.rda")
> print(plot(seizMix.bs, ordering = "coef.(Intercept)", col = Colors))
```

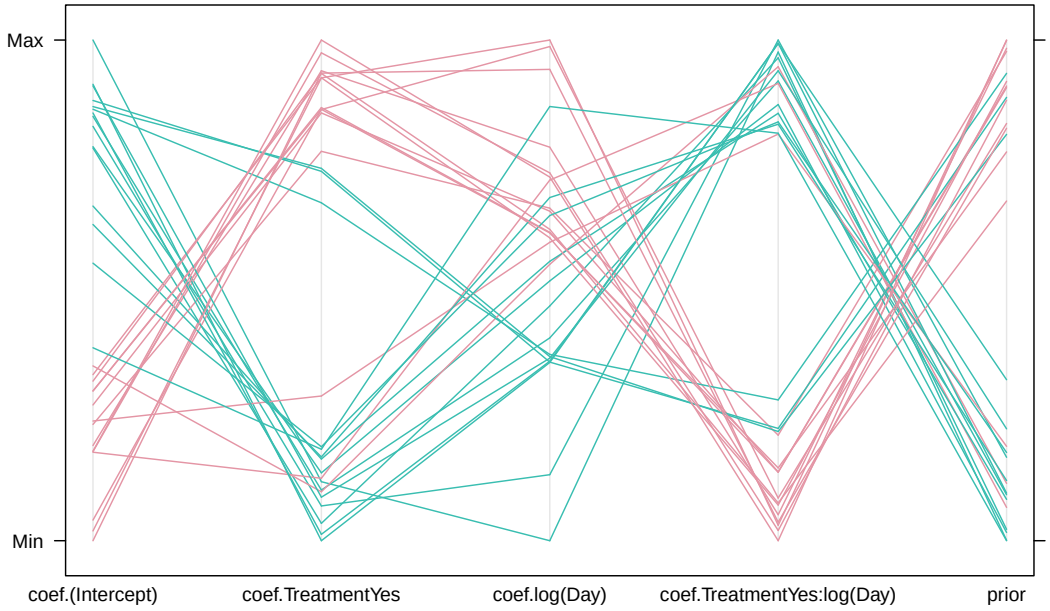

Figure 4: Diagnostic plot of the bootstrap results for the `seizure` data.

(where the ordering condition was imposed on) and again the trend. For all other parameter estimates unimodality is rejected which indicates that the ordering constraint was able to impose a unique labelling only for the own parameter and not for the other parameters. This suggests identifiability problems.

A. Generation of saved R output

```
> set.seed(123)
> ArtEx.bs <- boot(ArtEx.fit, R = 200, sim = "parametric")
> set.seed(123)
> seizMix.bs <- boot(seizMix, R = 200, sim = "parametric")
> save(ArtEx.bs, seizMix.bs, file = "boot-output.rda")
```

References

Canty A, Ripley B (2010). *boot: Bootstrap R (S-Plus) Functions*. R package version 1.2-43, URL http://CRAN.R-project.org/package=boot.

Davison AC, Hinkley DV (1997). *Bootstrap Methods and Their Application*. Cambridge Series on Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, UK. ISBN 0-521-57391-2 (hardcover), 0-521-57471-4 (paperback).

Grün B, Leisch F (2010). "Finite Mixture Model Diagnostics Using Resampling Methods." Unpublished manuscript.

Hennig C (2000). "Identifiability of Models for Clusterwise Linear Regression." *Journal of Classification*, **17**(2), 273–296.

Wang P, Puterman ML, Cockburn IM, Le ND (1996). "Mixed Poisson Regression Models with Covariate Dependent Rates." *Biometrics*, **52**, 381–400.

Affiliation:

Bettina Grün
Institut für Angewandte Statistik
Johannes Kepler Universität Linz
Altenbergerstraße 69
4040 Linz, Austria
E-mail: Bettina.Gruen@jku.at

Friedrich Leisch
Institut für Angewandte Statistik und EDV
Universität für Bodenkultur Wien
Peter Jordan Straße 82
1190 Wien, Austria
E-mail: Friedrich.Leisch@boku.ac.at
URL: http://www.statistik.lmu.de/~leisch/