name `readEViews()`. With this function, we can read in the data set from the Eviews file as follows.

```
> readEViews(hexViewFile("data4-1.wf1"))
```

```
Skipping boilerplate variable
Skipping boilerplate variable
   BATHS BEDRMS PRICE SQFT
1   1.75      3 199.9 1065
2   2.00      3 228.0 1254
3   2.00      3 235.0 1300
4   2.50      4 285.0 1577
5   2.00      3 239.0 1600
6   2.00      4 293.0 1750
7   2.75      4 285.0 1800
8   2.00      4 365.0 1870
9   2.50      4 295.0 1935
10  2.00      4 290.0 1948
11  3.00      4 385.0 2254
12  2.50      3 505.0 2600
13  3.00      4 425.0 2800
14  3.00      4 415.0 3000
```

This solution is not the most efficient way to read Eviews files, but the **hexView** package does make it easy to gradually build up a solution, it makes it easy to view the results, and it does provide a way to solve the problem without having to resort to C code.

Summary

The **hexView** package provides functions for viewing the raw byte contents of files. This is useful for exploring a file structure and for demonstrating how information is stored on a computer. More advanced functions make it possible to read quite complex binary formats using only R code.

Acknowledgements

At the heart of the **hexView** package is the `readBin()` function and the core facilities for working with `"raw"` binary objects in R code (e.g., `rawToChar()`); thanks to the R-core member(s) who were responsible for developing those features.

I would also like to thank the anonymous reviewer for useful comments on early drafts of this article.

Bibliography

IEEE Standard 754 for Binary Floating-Point Arithmetic. IEEE computer society, 1985. 4

R Development Core Team. *R Internals*. R Foundation for Statistical Computing, Vienna, Austria, 2006. URL `http://www.R-project.org`. ISBN 3-900051-14-3. 5

R. Ramanathan. *INTRODUCTORY ECONOMETRICS WITH APPLICATIONS*. Harcourt College, 5 edition, 2002. ISBN 0-03-034342-9. 6

Wikipedia. External data representation — wikipedia, the free encyclopedia, 2006a. URL `http://en.wikipedia.org/w/index.php?title=External_Data_Representation&oldid=91734878`. [Online; accessed 3-December-2006]. 4

Wikipedia. IEEE floating-point standard — wikipedia, the free encyclopedia, 2006b. URL `http://en.wikipedia.org/w/index.php?title=IEEE_floating-point_standard&oldid=89734307`. [Online; accessed 3-December-2006]. 4

Paul Murrell
Department of Statistics
The University of Auckland
New Zealand
`paul@stat.auckland.ac.nz`

FlexMix: An R Package for Finite Mixture Modelling

by Bettina Grün and Friedrich Leisch

Introduction

Finite mixture models are a popular method for modelling unobserved heterogeneity or for approximating general distribution functions. They are applied in a lot of different areas such as astronomy, biology, medicine or marketing. An overview on these models with many examples for applications is given in the recent monographs McLachlan and Peel (2000) and Frühwirth-Schnatter (2006).

Due to this popularity there exist many (standalone) software packages for finite mixture modelling (see McLachlan and Peel, 2000; Wedel and Kamakura, 2001). Furthermore, there are several different R packages for fitting finite mixture models available on CRAN. Packages which use the EM algorithm for model estimation are **flexmix**, **fpc**, **mclust**, **mixreg**, **mixtools**, and **mmlcr**. Packages with other model estimation methods are **bayesmix**, **depmix**, **moc**, **vabayelMix** and **wle**. A short description of these packages can be found in the CRAN task view on clustering (`http://cran.at.r-project.org/src/contrib/Views/Cluster.html`).

Finite mixture models

A finite mixture model is given by a convex combination of K different components, i.e. the weights of the components are non-negative and sum to one. For each component it is assumed that it follows a parametric distribution or is given by a more complex model, such as a generalized linear model (GLM).

In the following we consider finite mixture densities $h(\cdot|\cdot)$ with K components, dependent variables $\boldsymbol{y}$ and (optional) independent variables $\boldsymbol{x}$:

$$h(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w},\Theta) = \sum_{k=1}^{K} \pi_k(\boldsymbol{w},\boldsymbol{\alpha}) f(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\vartheta}_k)$$

where $\forall \boldsymbol{w},\boldsymbol{\alpha}$:

$$\pi_k(\boldsymbol{w},\boldsymbol{\alpha}) \geq 0 \,\forall k \quad \wedge \quad \sum_{k=1}^{K} \pi_k(\boldsymbol{w},\boldsymbol{\alpha}) = 1$$

and

$$\boldsymbol{\vartheta}_k \neq \boldsymbol{\vartheta}_l \quad \forall k \neq l.$$

We assume that the component distributions $f(\cdot|\cdot)$ are from the same distributional family with component specific parameters $\boldsymbol{\vartheta}_k$. The component weights or prior class probabilities π_k optionally depend on the concomitant variables $\boldsymbol{w}$ and the parameters $\boldsymbol{\alpha}$ and are modelled through multinomial logit models as suggested for example in Dayton and Macready (1988). A similar model class is also described in McLachlan and Peel (2000, p. 145). The model can be estimated using the EM algorithm (see Dempster et al., 1977; McLachlan and Peel, 2000) for ML estimation or using MCMC methods for Bayesian analysis (see for example Frühwirth-Schnatter, 2006).

A possible extension of this model class is to either have mixtures with components where the parameters of one component are fixed a-priori (e.g. zero-inflated models; Grün and Leisch, 2007b) or to even allow different component specific models (e.g. for modelling noise in the data; Dasgupta and Raftery, 1998).

Design principles of FlexMix

The main reason for the implementation of the package was to allow easy extensibility and to have the possibility for rapid prototyping in order to be able to try out new mixture models. The package was implemented using S4 classes and methods.

The EM algorithm provides a common basis for estimation of a general class of finite mixture models and the package **flexmix** tries to enable the user to exploit this commonness. **flexmix** provides the E-step and takes care of all data handling while the user is supposed to supply the M-step via model drivers for the component-specific model and the concomitant variable model. For the M-step available functions for weighted maximum likelihood estimation can be used as for example `glm()` for fitting GLMs or `multinom()` in **MASS** for multinomial logit models.

Currently model drivers are available for model-based clustering of multivariate Gaussian distributions with diagonal or unrestricted variance-covariance matrices (`FLXMCmvnorm()`) and multivariate Bernoulli and Poisson distributions (`FLXMCmvbinary()` and `FLXMCmvpois()`) where the dimensions are mutually independent. **flexmix** does not provide functionality for estimating mixtures of Gaussian distributions with special variance-covariance structures, as this functionality has already been implemented in the R package **mclust** (Fraley and Raftery, 2006).

For mixtures of regressions the Gaussian, binomial, Poisson and gamma distribution can be specified (`FLXMRglm()`). If some parameters are restricted to be equal over the components the model driver `FLXMRglmfix()` can be used. Zero-inflated Poisson and binomial regression models can be fitted using `FLXMRziglm()`. For an example of zero-inflated models see `example("FLXMRziglm")`. For the concomitant variable models either constant component weights (default) can be used or multinomial logit models (`FLXPmultinom()`) can be fitted.

Estimation problems can occur if the components become too small during the EM algorithm. In order to avoid these problems a minimum size can be specified for each component. This is especially important for finite mixtures of multivariate Gaussian distributions where full variance-covariance matrices are estimated for each component.

Further details on the implementation and the design principles as well as exemplary applications of the package can be found in the accompanying vignettes `"flexmix-intro"` which is an updated version of Leisch (2004) and `"regression-examples"` and in Grün and Leisch (2007a). Note that this article uses the new version 2.0 of the package, where the names of some driver functions have changed compared with older versions of **flexmix**.

Exemplary applications

In the following we present two examples for using the package. The first example demonstrates model-based clustering, i.e., mixtures without independent variables, and the second example gives an application for fitting mixtures of generalized linear regression models.

Model-based clustering

The following dataset is taken from Edwards and Allenby (2003) who refer to the Simmons Study of Media and Markets. It contains all households which used any whiskey brand during the last year and provides a binary incidence matrix on their brand use for 21 whiskey brands during this year. This means only the information on the different brands used in a household is available.

We first load the package and the dataset. The `whiskey` dataset contains observations from 2218 households. The relative frequency of usage for each brand is given in Figure 1. Additional information is available for the brands indicating the type of whiskey: blend or single malt.

```
R> library("flexmix")
R> data("whiskey")
R> set.seed(1802)
```

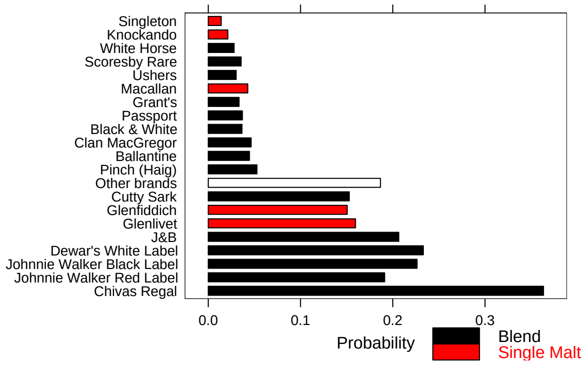

Figure 1: Relative frequency of the whiskey brands.

We fit a mixture of binomial distributions to the dataset where the variables in each component specific models are assumed to be independent. The EM algorithm is repeated `nrep = 3` times using random initialization, i.e. each observation is assigned to one component with an a-posteriori probability of 0.9 and 0.1 otherwise and the component is selected with equal probability.

```
R> wh_mix <- stepFlexmix(Incidence ~ 1,
+   weights = ~ Freq, data = whiskey,
+   model = FLXMCmvbinary(truncated = TRUE),
+   control = list(minprior = 0.005),
+   k = 1:7, nrep = 3)
```

Model-based clustering uses no explanatory variables, hence the right hand side of the formula `Incidence ~ 1` is constant. The model driver is `FLXMCmvbinary()` with argument `truncated = TRUE`, as the number of non-users is not available and a truncated likelihood is maximized in each M-step again using the EM-algorithm. We vary the number of components for `k = 1:7`. The best solution with respect to the log-likelihood for each of the different numbers of components is returned in an object of class `"stepFlexmix"`. The control argument can be used to control the fitting with the EM algorithm. With `minprior` the minimum relative size of the components is specified, components falling below this threshold are removed during the EM algorithm.

The dataset contains only the unique binary patterns observed with the corresponding frequency. We use these frequencies for the weights argument instead of transforming the dataset to have one row for each observation. The use of a weights argument allows to use only the number of unique observations for fitting, which can substantially reduce the size of the model matrix and hence speed up the estimation process. For this dataset this means that the model matrix has 484 instead of 2218 rows.

Model selection can be made using information criteria, as for example the BIC (see Fraley and Raftery, 1998). For this example the BIC suggests a mixture with 5 components:

```
R> BIC(wh_mix)

      1       2       3       4
27705.1 26327.6 25987.7 25683.2
      5       6       7
25647.0 25670.3 25718.6

R> wh_best <- getModel(wh_mix, "BIC")
R> wh_best

Call:
stepFlexmix(Incidence ~ 1,
    weights = ~Freq, data = whiskey,
    model = FLXMCmvbinary(truncated = TRUE),
    control = list(minprior = 0.005),
    k = 5, nrep = 3)

Cluster sizes:
  1   2   3   4   5
283 791 953  25 166

convergence after 180 iterations
```

The estimated parameters can be inspected using accessor functions such as `prior()` or `parameters()`.

```
R> prior(wh_best)

[1] 0.1421343 0.3303822
[3] 0.4311072 0.0112559
[5] 0.0851203

R> parameters(wh_best, component=4:5)[1:2,]

                     Comp.4
center.Singleton 0.643431
center.Knockando 0.601124
                       Comp.5
center.Singleton 2.75013e-02
center.Knockando 1.13519e-32
```

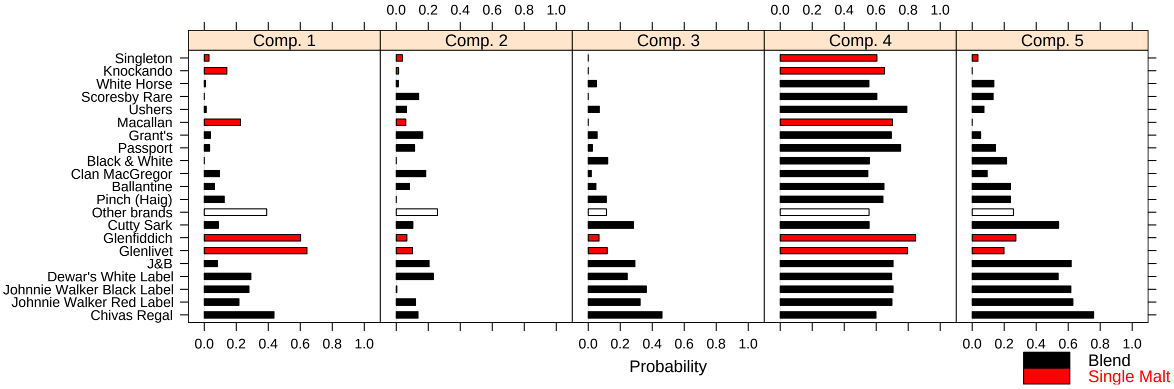

Figure 2: Estimated probability of usage for the whiskey brands for each component.

The fitted parameters of the mixture for each component are given in Figure 2. It can be seen that component 4 (1.1% of the households) contains the households which bought the greatest number of different brands and all brands to a similar extent. Households from component 5 (8.5%) also buy a wide range of whiskey brands, but tend to avoid single malts. Component 3 (43.1%) has a similar usage pattern as component 5 but buys less brands in general. Component 1 (14.2%) seems to favour single malt whiskeys and component 2 (33%) is especially fond of other brands and tends to avoid Johnnie Walker Black Label.

Mixtures of regressions

The patent data given in Wang et al. (1998) includes 70 observations on patent applications, R&D spending and sales in millions of dollar from pharmaceutical and biomedical companies in 1976 taken from the National Bureau of Economic Research R&D Masterfile. The data is given in Figure 3.

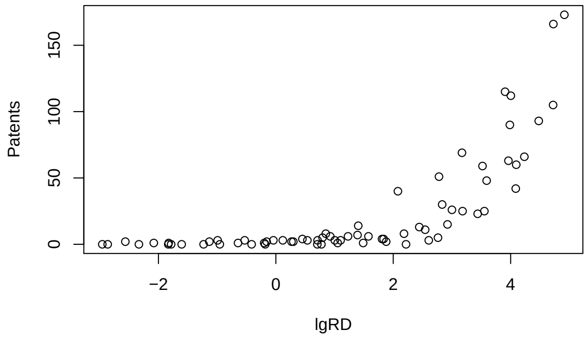

Figure 3: Patent dataset.

The model which is chosen as the best in Wang et al. (1998) is a finite mixture of three Poisson regression models with `Patents` as dependent variable, the logarithmized R&D spending `lgRD` as independent variable and the R&D spending per sales `RDS` as concomitant variable. This model can be fitted in R with the component-specific model driver `FLXMRglm()` which allows fitting of finite mixtures of GLMs. As concomitant variable model driver `FLXPmultinom()` is used for a multinomial logit model where the posterior probabilities are the dependent variables.

```
R> data("patent")
R> pat_mix <- flexmix(Patents ~ lgRD,
+   k = 3, data = patent,
+   model = FLXMRglm(family = "poisson"),
+   concomitant = FLXPmultinom(~RDS))
```

The observed values together with the fitted values for each component are given in Figure 4. The coloring and characters used for plotting the observations are according to the component assignment using the maximum a-posteriori probabilities, which are obtained using `cluster(pat_mix)`.

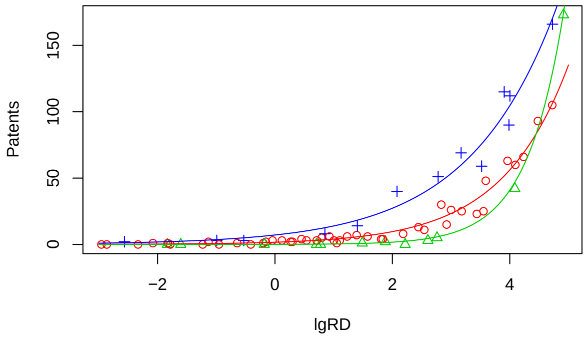

Figure 4: Patent data with fitted values for each component.

In Figure 5 a rootogram of the posterior probabilities of the observations is given. This is the default plot of the `"flexmix"` objects returned by the fitting function. It can be used for arbitrary mixture models and indicates how well the observations are clustered by the mixture. For ease of interpretation the observations with a-posteriori probability less than eps=10^{-4} are omitted as otherwise the peak at zero would dominate the plot. The observations where the a-posteriori probability is largest for the third component are colored differently. The plot is generated using the following command.

```
R> plot(pat_mix, mark = 3)
```

The posteriors of all three components have modes at 0 and 1, indicating well-separated clusters (Leisch, 2004). Note that the object returned by the plot function is of class "trellis", and that the plot itself is produced by the corresponding show() method (Sarkar, 2002).

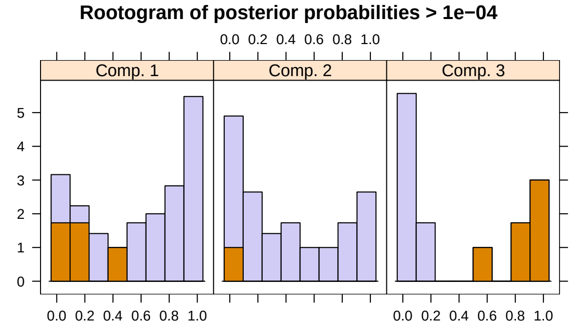

Figure 5: Rootogram of the posterior probabilities.

Further details of the fitted mixture can be obtained with refit() which returns the fitted values together with approximate standard deviations and significance tests, see Figure 6. The standard deviations are only approximative because they are determined separately for each component and it is not taken into account that the components have been estimated simultaneously. In the future functionality to determine the standard deviations using either the full Hesse matrix or the parametric bootstrap shall be provided.

The estimated coefficients are given in Figure 7. The black lines indicate the (approximative) 95% confidence intervals. This is the default plot for the objects returned by refit() and is obtained with the following command.

```
R> plot(refit(pat_mix), bycluster = FALSE)
```

The argument bycluster indicates if the clusters/components or the different variables are used as conditioning variables for the panels.

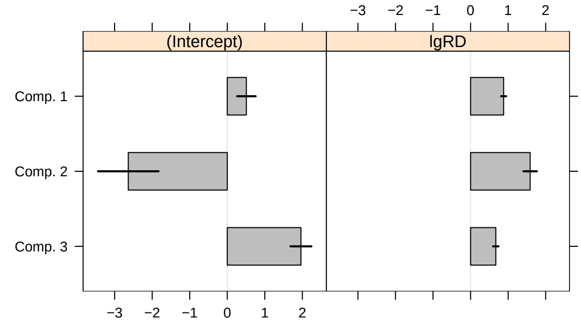

Figure 7: Estimated coefficients of the component specific models with corresponding 95% confidence intervals.

The plot indicates that the estimated coefficients vary between all components even though the coefficients for lgRD are similar for the first and third component. A smaller model where these coefficients are restricted to be equal can be fitted using the model driver FLXMRglmfix(). The EM algorithm can be initialized in the original solution using the estimated posterior probabilities for the cluster argument. As in this case the first and third component are restricted to have the same coefficient for lgRD, the posteriors of the fitted mixture are used for initialization after reordering the components to have these two components next to each other. The modified model is compared to the original model using the BIC.

```
R> Model_2 <- FLXMRglmfix(family = "poisson",
+   nested = list(k = c(1,2),
+     formula = ~lgRD))
R> Post_1 <- posterior(pat_mix)[,c(2,1,3)]
R> pat_mix2 <- flexmix(Patents ~ 1,
+   concomitant = FLXPmultinom(~RDS),
+   data = patent, cluster = Post_1,
+   model = Model_2)
R> c(M_1 = BIC(pat_mix), M_2 = BIC(pat_mix2))

    M_1     M_2
437.836 445.243
```

In this example, the original model is preferred by the BIC.

Summary

flexmix provides infrastructure for fitting finite mixture models with the EM algorithm and tools for model selection and model diagnostics. We have shown the application of the package for model-based clustering as well as for fitting finite mixtures of regressions.

In the future we want to implement new model drivers, e.g., for generalized additive models with smooth terms, as well as to extend the tools for model selection, diagnostics and model validation. Additional functionality will be added which allows to fit mixture models with different component specific models. The implementation of zero-inflated models has been a first step in this direction.

Acknowledgments

This research was supported by the Austrian Science Foundation (FWF) under grant P17382.

Bibliography

A. Dasgupta and A. E. Raftery. Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American Statistical Association*, 93(441):294–302, March 1998. 9

```
R> refit(pat_mix)

Call:
refit(pat_mix)

Number of components: 3

$Comp.1
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.50819    0.12366    4.11 3.96e-05 ***
lgRD         0.87976    0.03328   26.43  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

$Comp.2
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.6368     0.4043  -6.522 6.95e-11 ***
lgRD          1.5866     0.0899  17.648  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

$Comp.3
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.96217    0.13968   14.05   <2e-16 ***
lgRD         0.67190    0.03572   18.81   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 6: Details of the fitted mixture model for the patent data.

C. M. Dayton and G. B. Macready. Concomitant-variable latent-class models. *Journal of the American Statistical Association*, 83(401):173–178, March 1988. 9

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM-algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977. 9

Y. D. Edwards and G. M. Allenby. Multivariate analysis of multiple response data. *Journal of Marketing Research*, 40:321–334, August 2003. 10

C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8): 578–588, 1998. 10

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, September 2006. 9

S. Frühwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer Series in Statistics. Springer, New York, 2006. ISBN 0-387-32909-9. 8, 9

B. Grün and F. Leisch. Fitting finite mixtures of generalized linear regressions in R. *Computational Statistics & Data Analysis*, 2007a. Accepted for publication. 9

B. Grün and F. Leisch. Flexmix 2.0: Finite mixtures with concomitant variables and varying and fixed effects. *Submitted for publication*, 2007b. 9

F. Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8), 2004. URL `http://www.jstatsoft.org/v11/i08/`. 9, 12

G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000. 8, 9

D. Sarkar. Lattice. *R News*, 2(2):19–23, June 2002. URL `http://CRAN.R-project.org/doc/Rnews/`. 12

P. Wang, I. M. Cockburn, and M. L. Puterman. Analysis of patent data — A mixed-Poisson-regression-model approach. *Journal of Business & Economic Statistics*, 16(1):27–41, 1998. 11

M. Wedel and W. A. Kamakura. *Market Segmentation — Conceptual and Methodological Foundations*. Kluwer Academic Publishers, second edition, 2001. ISBN 0-7923-8635-3. 8

Bettina Grün
Technische Universität Wien, Austria
`Bettina.Gruen@ci.tuwien.ac.at`

Friedrich Leisch
Ludwig-Maximilians-Universität München, Germany
`Friedrich.Leisch@R-project.org`