Using Weights in mclust

Thomas Brendan Murphy
University College Dublin, Ireland
brendan.murphy@ucd.ie

Luca Scrucca
Università degli Studi di Perugia, Italy
luca@stat.unipg.it

June 2012

1 Introduction

The mclust package (Fraley and Raftery, 2002, 2006) can be extended to allow for each observation to have a weight which controls its contribution to the model fit. A brief background on how weights can be included is given in Section 2. A function `me.weighted()`, which is included in mclust, provides a weighted version of the `me()` function in mclust. The use of `me.weighted()` is demonstrated on two examples in Section 3 and Section 4.

2 Background

The log-likelihood function l and complete-data log-likelihood function l_c for the finite mixture of normal distributions, as used in mclust, are of the form

$$l = \sum_{i=1}^{n} \log \left[\sum_{g=1}^{G} \tau_g f(\mathbf{x}_i | \mu_g, \Sigma_g) \right] \tag{1}$$

and

$$l_c = \sum_{i=1}^{n} \sum_{g=1}^{G} z_{ig} \log \left[\tau_g f(\mathbf{x}_i | \mu_g, \Sigma_g) \right]. \tag{2}$$

Suppose we introduce weights for each observation, so that the weight for observation i is given by w_i, then we can define the log-likelihood $l^{(w)}$ and complete-data log-likelihood $l_c^{(w)}$ to have the form

$$l^{(w)} = \sum_{i=1}^{n} w_i \log \left[\sum_{g=1}^{G} \tau_g f(\mathbf{x}_i | \mu_g, \Sigma_g) \right] \tag{3}$$

and

$$l_c^{(w)} = \sum_{i=1}^{n} \sum_{g=1}^{G} z_{ig} w_i \log \left[\tau_g f(\mathbf{x}_i | \mu_g, \Sigma_g) \right]. \tag{4}$$

The E-step of the EM algorithm for fitting the normal mixture model with weights is unchanged by the introduction of the weights. However, the M-step is changed in that the values z_{ig} in the

complete-data log-likelihood function (2) are replaced by $z_{ig}w_i$ in the weighted complete-data log-likelihood (4). Thus, if these changed values are supplied into the `mstep()` function (and related functions) in mclust then the weights can be used in an appropriate manner for model fitting. The `me.weighted()` function has been written to implement the EM algorithm for fitting normal mixtures with weights where the first step of the algorithm is an M-step; the function is similar in syntax to the `me()` function. The function takes the same arguments as `me()` but it also takes an optional additonal vector of weights. Note that, if any of the weights are greater than one, then all of the weights are rescaled to have a maximum of one. Also, the parameter estimates are not effected by multiplying the weights by a constant value.

3 Example: Parameter Sensitivity

We can use mclust to complete a model-based clustering of the iris data (Anderson, 1935). However, we may wish to assess the sensitivity of the fit to the deletion of an observation from the data. This can be easily achieved by assigning the deleted observation, j, a weight $w_j = 0$ and each other observations weights $w_i = 1$ for $i \neq j$.

First, we use the `Mclust` function to complete a model-based clustering of the iris data. The output is stored in the object `fit`.

```
> data(iris)
> X <- iris[,-5]
> fit <- Mclust(X)
```

We now want to look at the impact of deleting an observation on the fitted parameter estimates. We delete observation $j = 1$ and compare the original and new parameter estimates.

```
> j <- 1
> N <- nrow(iris)
> w <- rep(1,N)
> w[j] <- 0
> fitnew <- do.call("me.weighted",c(list(data=X,weights=w),fit))
```

The original and new component probabilities are:

```
> print(fit$parameters$pro)
[1] 0.33333 0.66667
> print(fitnew$parameters$pro)
[1] 0.32886 0.67114
```

The original and new component means are:

```
> print(fit$parameters$mean)
              [,1]  [,2]
Sepal.Length 5.006 6.262
Sepal.Width  3.428 2.872
Petal.Length 1.462 4.906
Petal.Width  0.246 1.676
> print(fitnew$parameters$mean)
```

```
               [,1]  [,2]
Sepal.Length 5.00408 6.262
Sepal.Width  3.42654 2.872
Petal.Length 1.46327 4.906
Petal.Width  0.24694 1.676
```

Finally, the original and new component covariance parameters are:

```
> print(fit$parameters$variance$sigma)
, , 1

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length     0.150651    0.130801    0.0208446   0.0130910
Sepal.Width      0.130801    0.176045    0.0160325   0.0122145
Petal.Length     0.020845    0.016032    0.0280826   0.0060157
Petal.Width      0.013091    0.012215    0.0060157   0.0104237

, , 2

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length      0.40004    0.108654      0.39940    0.143682
Sepal.Width       0.10865    0.109281      0.12389    0.072844
Petal.Length      0.39940    0.123890      0.61090    0.257389
Petal.Width       0.14368    0.072844      0.25739    0.168082
> print(fitnew$parameters$variance$sigma)
, , 1

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length     0.153558    0.133385    0.0214014   0.0134660
Sepal.Width      0.133385    0.179578    0.0165283   0.0125735
Petal.Length     0.021401    0.016528    0.0285980   0.0060943
Petal.Width      0.013466    0.012573    0.0060943   0.0106006

, , 2

             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length      0.40066    0.108856      0.40018    0.144000
Sepal.Width       0.10886    0.109222      0.12420    0.072928
Petal.Length      0.40018    0.124198      0.61194    0.257876
Petal.Width       0.14400    0.072928      0.25788    0.168274
```

Thus, we can see the impact of deleting a single observation from the data. The above code can be easily adapted to consider deleting data subsets of size greater than one.

4 Example: Bootstrapping

The inclusion of weights also facilitates computing bootstrap standard errors for parameter estimates; the use of boostrapping (Efron, 1979; Efron and Tibshirani, 1993) to compute standard errors for mclust parameter estimates is explored in Everitt and Hothorn (2010), O'Hagan (2012) and O'Hagan et al. (2012). The following code chunk uses B bootstrap samples to find standard errors for the mclust model parameters.

```
> library(mclust)
> data(iris)
> X <- iris[,-5]
> N <- nrow(X)
> fit <- Mclust(X)
> z <- fit$z
> G <- fit$G
> M <- ncol(X)
> B <- 100
> pro.boot <- matrix(NA,B,G)
> mu.boot <- array(NA,c(B,M,G))
> sigma.boot <- array(NA,c(B,M,M,G))
> for (b in 1:B)
  {
    ind <- sample(1:N,size=N,replace=TRUE)
    w <- rep(0,N)
    names(w) <- 1:N
    tab <- table(ind)
    w[names(tab)] <- tab
    w <- w/max(w)
    fit.boot <- me.weighted(fit$modelName,X,z,w)
    pro.boot[b,] <- fit.boot$parameters$pro
    mu.boot[b,,] <- fit.boot$parameters$mean
    sigma.boot[b,,,] <- fit.boot$parameters$variance$sigma
  }
```

The objects `pro.boot`, `mu.boot`, `sigma.boot` store the parameter estimates for each bootstrap iteration and the standard errors for the parameters can be computed from the stored bootstrap parameter estimates.

We get the following standard errors for the mixture component probabilities:

```
> apply(pro.boot,2,sd)
[1] 0.037274 0.037274
```

We get the following standard errors for the mixture component means:

```
> apply(mu.boot,c(2,3),sd)
         [,1]     [,2]
[1,] 0.051907 0.066895
[2,] 0.054735 0.036123
[3,] 0.023487 0.084301
[4,] 0.013252 0.041707
```

We get the following standard errors for the covariance parameters:

```
> apply(sigma.boot,c(2,3,4),sd)
, , 1

          [,1]      [,2]      [,3]      [,4]
[1,] 0.0249235 0.0205639 0.0139814 0.0058518
[2,] 0.0205639 0.0331674 0.0121922 0.0066415
[3,] 0.0139814 0.0121922 0.0071221 0.0030908
[4,] 0.0058518 0.0066415 0.0030908 0.0024303
```

```
, , 2

         [,1]     [,2]     [,3]     [,4]
[1,] 0.048735 0.025538 0.062782 0.026140
[2,] 0.025538 0.017577 0.034146 0.017261
[3,] 0.062782 0.034146 0.081047 0.031755
[4,] 0.026140 0.017261 0.031755 0.018779
```

5 Conclusions

The use of weights can be facilitated in mclust by carefully editing existing code. A function called `me.weighted()` has been developed to facilitate the inclusion of weights in model-based clustering. The facility to include weights can be used to assess the mclust fit and model outputs more deeply.

References

Anderson, E. (1935), "The irises of the Gapse Peninsula," *Bulletin of the American Iris Society*, 59, 2–5.

Efron, B. (1979), "Bootstrap methods: another look at the jackknife," *Ann. Statist.*, 7, 1–26.

Efron, B. and Tibshirani, R. J. (1993), *An introduction to the bootstrap*, vol. 57 of *Monographs on Statistics and Applied Probability*, New York: Chapman and Hall.

Everitt, B. and Hothorn, T. (2010), *A handbook of statistical analyses using R*, Boca Raton: Chapman & Hall/CRC.

Fraley, C. and Raftery, A. E. (2002), "Model-based Clustering, Discriminant Analysis and Density Estimation," *Journal of the American Statistical Association*, 97, 611–631.

— (2006), "MCLUST Version 3 for R: Normal Mixture Modeling and Model-based Clustering." Tech. Rep. 504, Department of Statistics, University of Washington, revised 2009.

O'Hagan, A. (2012), "Topics in model-based clustering and classification," Ph.D. thesis, University College Dublin, Ireland.

O'Hagan, A., Gormley, I. C., and Murphy, T. B. (2012), "Bootstrap and Jackknife Methods of Standard Error Estimation in Model-Based Clustering," *In Preparation*.