MCLUST: Software for Model-Based Clustering, Density Estimation and Discriminant Analysis *†

Chris Fraley and Adrian E. Raftery
Technical Report No. 415
Department of Statistics
University of Washington
Box 354322
Seattle, WA 98195-4322 USA

October 1, 2002

ΜCLUST is a software package for model-based clustering, density estimation and discriminant analysis interfaced to the S-PLUS commercial software. 1 2 It implements parameterized Gaussian hierarchical clustering algorithms and the EM algorithm for parameterized Gaussian mixture models with the possible addition of a Poisson noise term. Also included are functions that combine hierarchical clustering, EM and the Bayesian Information Criterion (BIC) in comprehensive strategies for clustering, density estimation, and discriminant analysis. ΜCLUST provides functionality for displaying and visualizing clustering and classification results. A web page with related links can be found at

http://www.stat.washington.edu/mclust

---

* Funded by Office of Naval Research contracts N00014-96-1-0192, N00014-96-1-0330, and N00014-97-1-0736, and by National Institutes of Health grant 8 R01 EB002137-02
† Thanks go to Ron Wehrens for comments and suggestions leading to improvements in the ΜCLUST documentation and software.
1 Insightful Corp., Seattle, WA USA — http://www.insightful.com/splus
2 ΜCLUST has been ported to the R language by Ron Wehrens — see http://lib.stat.cmu.edu/R/CRAN.
## Contents

1 Models .......................................................... 4

2 Obtaining and Installing MCLUST ........................................ 5
   2.1 Using MCLUST with S-PLUS 6 for UNIX/Linux .......................... 5
   2.2 Using MCLUST with S-PLUS 6 for Windows ................................ 6

3 Hierarchical Clustering ............................................. 6

4 EM for Mixture Models ............................................. 8
   4.1 Uncertainty ........................................................................ 9
   4.2 Individual E and M Steps .................................................. 9

5 Bayesian Information Criterion ........................................ 10

6 Cluster Analysis .......................................................... 11
   6.1 Mclust ........................................................................... 11
   6.2 Emclust ........................................................................... 14
   6.3 Clustering with Noise and Outliers ...................................... 17

7 Simulation from Mixture Densities ..................................... 19

8 Density Estimation .......................................................... 21

9 Displays ...................................................................... 22
   9.1 Plotting Two-Dimensional Results ..................................... 22
   9.2 Plotting Multidimensional Results ...................................... 24
      9.2.1 Coordinate Projections ............................................. 24
      9.2.2 Random Projections ................................................ 29

10 Discriminant Analysis .................................................... 31
    10.1 Discriminant Analysis using mstep and estep ....................... 32
    10.2 Mixture Discriminant Analysis via MclustDA ....................... 33
    10.2.1 mclustDA ................................................................ 34
    10.2.2 mclustDAtain and mclustDAtest ................................. 34

11 One Dimensional Data .................................................... 38
   11.1 Clustering ................................................................... 38
   11.2 Discriminant Analysis .................................................. 38
   11.3 "mclust" option for S-PLUS density function ....................... 41

12 Extensions .................................................................... 42
   12.1 Large Data Sets .......................................................... 42
   12.2 High Dimensional Data ................................................ 42

13 Function Summary ........................................................ 43
   13.1 Hierarchical Clustering .................................................. 43
   13.2 Parameterized Gaussian Mixture Models .............................. 43
   13.3 Density Computation for Parameterized Gaussian Mixtures ...... 43
   13.4 Model-based Clustering / Density Estimation ....................... 43
   13.5 Discriminant Analysis .................................................. 43
   13.6 Support for Modeling and Classification ............................. 44
   13.7 Plotting Functions ........................................................ 44
      13.7.1 One-Dimensional Data ............................................. 44
      13.7.2 Two-Dimensional Data ............................................. 44
List of Tables

1  Parameterizations of $\Sigma_k$ currently available in MCLUST for multidimensional data.  

List of Figures

1  Pairs plot of Fisher’s iris data showing species.  
2  Uncertainty plot for model-based classification from EM for Fisher’s iris data.  
3  Plots associated with the mclust function.  
4  BIC values from EMclust applied to Fisher’s iris data.  
5  Cluster analysis with noise.  
6  Data simulated from models of Fisher’s iris data.  
7  Lansing Woods maple trees.  
8  Density Estimation for the Lansing Woods maples.  
9  Classification and uncertainty plots for the Lansing Woods maples.  
10  Density and uncertainty surfaces for the Lansing Woods maples.  
11  Coordinate projections of a model-based classification for Fisher’s iris data.  
12  Random projections of a model-based classification for Fisher’s iris data.  
13  Lansing Woods hickory and maples trees.  
14  Discriminant analysis using mstep and estep.  
15  Plots associated with mclustDA.  
16  mclustDAtrain Models.  
17  Model-based clustering of one-dimensional data.  
18  Comparison of one-dimensional density estimates.
The MCLUST software, with offerings originally limited to model-based hierarchical clustering, was extended to include EM for parameterized Gaussian mixture models, as well as a clustering strategy in which the model and number of clusters are simultaneously selected via the Bayesian Information Criterion (BIC) [10]. This manuscript describes a substantial upgrade to MCLUST, which includes the following among its new features:

- EM for four diagonal covariance mixture models.
- Density estimation via parameterized Gaussian mixtures.
- Simulation from parameterized Gaussian mixtures.
- Discriminant analysis via MclustDA.
- Methods for one dimensional data.
- Enhanced displays, including uncertainty plots and random projections.

A comprehensive treatment of the methods used in MCLUST can be found in [11].

1 Models

In MCLUST, each cluster is represented by a Gaussian model

$$
\phi_k(x | \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\},
$$

where $x$ represents the data, and $k$ is an integer subscript specifying a particular cluster. Clusters are ellipsoidal, centered at the means $\mu_k$. The covariances $\Sigma_k$ determine their other geometric features.

Each covariance matrix is parameterized by eigenvalue decomposition in the form

$$
\Sigma_k = \lambda_k D_k A_k D_k^T,
$$

where $D_k$ is the orthogonal matrix of eigenvectors, $A_k$ is a diagonal matrix whose elements are proportional to the eigenvalues of $\Sigma_k$, and $\lambda_k$ is a scalar. The orientation of the principal components of $\Sigma_k$ is determined by $D_k$, while $A_k$ determines the shape of the density contours; $\lambda_k$ specifies the volume of the corresponding ellipsoid, which is proportional to $\lambda_k^d |A_k|$, where $d$ is the data dimension. Characteristics (orientation, volume and shape) of distributions are usually estimated from the data, and can be allowed to vary between clusters, or constrained to be the same for all clusters [17, 2, 6]. This parameterization includes but is not restricted to well-known models such as equal-volume spherical variance ($\Sigma_k = \lambda I$) which gives the sum of squares criterion [21], constant variance [12], and unconstrained variance [20].

In one dimension, there are just two models: $E$ for equal variance and $V$ for varying variance. In more than one dimension, the model identifiers code geometric characteristics of the model. For example, EVI denotes a model in which the volumes of all clusters are equal ($E$), the shapes of the clusters may vary ($V$), and the orientation is the identity ($I$).
Clusters in this model have diagonal covariances with orientation parallel to the coordinate axes. Parameters associated with characteristics designated by E or V are determined from the data. Table 1 shows the various multivariate model options currently available in MCLUST for hierarchical clustering (denoted HC) and EM.

Table 1: Parameterizations of the covariance matrix $\Sigma_k$ currently available in MCLUST for hierarchical clustering (HC) and/or EM for multidimensional data. (‘•’ indicates availability).

<table>
<thead>
<tr>
<th>identifier</th>
<th>Model</th>
<th>HC</th>
<th>EM</th>
<th>Distribution</th>
<th>Volume</th>
<th>Shape</th>
<th>Orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>E11</td>
<td>$\lambda I$</td>
<td>•</td>
<td>•</td>
<td>Spherical</td>
<td>equal</td>
<td>equal</td>
<td>NA</td>
</tr>
<tr>
<td>V11</td>
<td>$\lambda_k I$</td>
<td>•</td>
<td>•</td>
<td>Spherical</td>
<td>variable</td>
<td>equal</td>
<td>NA</td>
</tr>
<tr>
<td>E1E</td>
<td>$\lambda A$</td>
<td>•</td>
<td>•</td>
<td>Diagonal</td>
<td>equal</td>
<td>equal</td>
<td>coordinate axes</td>
</tr>
<tr>
<td>V1E</td>
<td>$\lambda_k A$</td>
<td>•</td>
<td>•</td>
<td>Diagonal</td>
<td>variable</td>
<td>equal</td>
<td>coordinate axes</td>
</tr>
<tr>
<td>V1V</td>
<td>$\lambda_k A_k$</td>
<td>•</td>
<td>•</td>
<td>Diagonal</td>
<td>variable</td>
<td>equal</td>
<td>coordinate axes</td>
</tr>
<tr>
<td>EEE</td>
<td>$\lambda DAD^T$</td>
<td>•</td>
<td>•</td>
<td>Ellipsoidal</td>
<td>equal</td>
<td>equal</td>
<td>equal</td>
</tr>
<tr>
<td>EEV</td>
<td>$\lambda D_kAD_k^T$</td>
<td>•</td>
<td>•</td>
<td>Ellipsoidal</td>
<td>equal</td>
<td>variable</td>
<td></td>
</tr>
<tr>
<td>VEV</td>
<td>$\lambda_k D_kAD_k^T$</td>
<td>•</td>
<td>•</td>
<td>Ellipsoidal</td>
<td>variable</td>
<td>equal</td>
<td>variable</td>
</tr>
<tr>
<td>VVV</td>
<td>$\lambda_k D_k A_k D_k^T$</td>
<td>•</td>
<td>•</td>
<td>Ellipsoidal</td>
<td>variable</td>
<td>variable</td>
<td>variable</td>
</tr>
</tbody>
</table>

2 Obtaining and Installing MCLUST

MCLUST can be obtained in one of several ways:

- via the world wide web at http://www.stat.washington.edu/mclust
- via anonymous ftp from ftp.u.washington.edu in the directory public/mclust
- from Statlib; see http://lib.stat.cmu.edu/S/mclust

Note: The S-PLUS density function has been changed to add a method = "mclust" option (see Section 11.3). In both UNIX/Linux and Windows, there will be a warning that density has been masked. However, the default behavior of density remains unaltered.

2.1 Using MCLUST with S-PLUS 6 for UNIX/Linux

The file MCLUST.tar.gz is a packed version of a directory containing all the necessary files for incorporating MCLUST into S-PLUS on UNIX systems. The commands to unpack it (and remove the tar file) are:

```bash
gunzip MCLUST.tar.gz
tar xvf MCLUST.tar
rm MCLUST.tar
```

This creates a directory called MCLUST. To run the software in this directory, do the following.
cd MCLUST
Splus CHAPTER
Splus make
Splus TRUNC_AUDIT 0

The ‘Splus make’ command

- compiles the Fortran code, which will automatically loaded when S-PLUS is invoked in this directory.
- sources the S-PLUS functions in mclust.q into the .Data directory
- compiles and loads the help files (file extension .sgml)

The software can now be used by running S-PLUS in directory MCLUST. They can also be run in a different working directory, using the following S-PLUS commands

> attach("./MCLUST/.Data")

where ./ indicates the path to the directory MCLUST.

2.2 Using MCLUST with S-PLUS 6 for Windows

The file Mclust.zip contains the necessary files for running MCLUST with S-PLUS 6 for Windows. To install MCLUST on Windows, extract the contents of Mclust.zip to a directory called mclust. That directory needs to be attached in S-PLUS in order to access the MCLUST functions. To use the MCLUST help files, either click on the file mclust.chm included in the folder, or else execute it as a command from a DOS prompt.

3 Hierarchical Clustering

MCLUST provides functions hc for model-based hierarchical agglomeration, and hclass for determining the resulting classifications. The function implements fast methods based on the multivariate normal classification likelihood [9]. As an example of the use of hc and hclass, consider Fisher’s iris data [8], which is available as a data set in S-PLUS.\(^3\) Figure 1 is a pairs plot of the iris data in which the three species are differentiated by symbol.

We first transform the data from a three-dimensional array to a matrix in which the species information is lost, then apply the hierarchical clustering algorithm for variable volume spherical variances (VII):

> irisMatrix <- matrix(aperm(iris,c(1,3,2)),150,4,dimn=dimnames(iris)[1:2])
> hcVIIiris <- hc(modelName = "VII", data = irisMatrix)

The classification produced by hc for various numbers of clusters can be obtained with hclass. For example, for the classifications corresponding to 2 and 3 clusters:

\(^3\)A description of this data set is available in S-PLUS from the command line via help(iris) or from the S-PLUS help window under ‘datasets’.
Figure 1: Pairs plot of Fisher’s iris data showing classification into species.

> cl <- hclass(hcVIIiris, 2:3)

The classifications can be displayed with the data using clPairs:

> clPairs(data = irisMatrix, classification = cl[,"2"])
> clPairs(data = irisMatrix, classification = cl[,"3"])

More options for displaying clustering and classification results are discussed in Section 9.

The function hc starts by default with every observation of the data in a cluster by itself, and continues until all observations are merged into a single cluster. Arguments partition and minclus can be used to initialize the process at a chosen nontrivial partition, and to stop it before it reaches the final stage of merging.
4 EM for Mixture Models

MCLUST provides iterative EM (Expectation-Maximization) methods for maximum likelihood clustering with parameterized Gaussian mixture models. In this application, an iteration of EM consists of an ‘E’-step, which computes a matrix $z$ such that $z_{ik}$ is an estimate of the conditional probability that observation $i$ belongs to group $k$ given the current parameter estimates, and an ‘M-step’, which computes maximum likelihood parameter estimates given $z$. In the limit, the parameters usually converge to the maximum likelihood values for the Gaussian mixture model

$$
\prod_{i=1}^{n} \sum_{k=1}^{G} \tau_k \phi_k(x_i \mid \mu_k, \Sigma_k),
$$

and columns means converge to the mixing proportions $\tau_k$, where $n$ is the number of observations in the data. Here $G$ is the number of groups, which is fixed in the EM algorithm. The parameterizations of $\Sigma_k$ currently available for EM in MCLUST are listed in Table 1. They are a subset of the parameterizations discussed in [6], which gives details of the EM algorithm for these models.

MCLUST functions `em`, `me` (iterated M-step followed by E-step), `estep` and `mstep` implement EM for the parameterized Gaussian mixtures. Given the data, an initial estimate of $z$, and the model specification, `me` produces the values of $z$ associated with maximum likelihood parameters. Initial estimates of $z$ may be obtained from a discrete classification, that is, a matrix of indicator variables with exactly one 1 per row. For example, `me` can be started with a classification produced by `hc`:

```r
> hcVVWiris <- hc( modelName = "VVV", data = irisMatrix) # unconstrained model
> cl <- hclass(hcVVWiris, 3) # 3-group hc classification
> meVVWiris <- me( modelName = "VVV", data = irisMatrix, z = unmap(cl))
```

The function `unmap` converts a discrete classification into the corresponding indicator variables. In general, the models used in `hc` and `me` need not be the same. It may in some cases be desirable to use one of the faster methods in `hc` (e.g. spherical or unconstrained models), followed by specification of a more complex model for EM.

For any matrix $z$ of conditional probabilities, there is a corresponding discrete classification assigning each observation to the group represented by the column in which the $z$ value for that observation (row) is maximized. MCLUST provides a function `map` for converting $z$ to this ‘nearest’ classification. The following call to `clPairs` plots the iris data along with its classification obtained from `me` in the above example:

```r
> clPairs( irisMatrix, map(meVVWiris$z))
```

Besides initial values, other parameters can influence the outcome of `em` or `me`. These include:

- `tol` Iteration convergence tolerance. The default is `Mclust$tol=c(1.e-5, 1.e-5)`, where the first value is the tolerance for relative convergence of the loglikelihood in the EM algorithm, and the second value is the relative parameter convergence tolerance for the M-step for those models that have an iterative M-step ("VEI", "VEE", "VWE", "VEV").
eps A tolerance for terminating iterations due to ill-conditioning, such as near singularity in covariance matrices. The default is \texttt{.Mclust$eps} which is the relative machine precision \texttt{2.220446e-16} in the release.

Although these are in some sense hidden by the defaults, they may have a significant effect on results and should be taken into consideration in analysis. An example in which \texttt{tol} = 1.e-4 and \texttt{tol} = 1.e-5 give quite different results is given in Section 8.

### 4.1 Uncertainty

The uncertainty in the classification associated with \( z \) can be obtained by subtracting the probability of the most likely group for each observation from 1:

\[
> \text{uncer} <- 1 - \text{apply( meVWiris$z, 1, max)}
\]

The S-PLUS function \texttt{quantile} applied to the uncertainty gives a measure of the quality of the classification.

\[
> \text{quantile(uncer)}
\]

\[
\begin{array}{cccc}
\text{0\%} & 25\% & 50\% & 75\% & 100\% \\
0 & 0.949809e-08 & 0.001384177 & 0.3378195
\end{array}
\]

In this case the indication is that the majority of observations are well classified. Note, however, that when groups intersect, uncertain classifications would be expected in the overlapping regions.

When a true classification is known, the relative uncertainty of misclassified observations can be displayed by the function \texttt{uncerPlot}, as is done below for the iris example (see Figure 2):

\[
> \text{irisClass <- rep(dimnames(iris)[[3]], rep(50,3))}
\]

\[
> \text{uncerPlot(z = meVWiris$z, truth = irisClass)}
\]

Plotting an uncertainty surface is also possible for two-dimensional data (see Section 9.1).

### 4.2 Individual E and M Steps

\texttt{MCLUST} includes functions \texttt{estep} and \texttt{mstep} implementing the individual steps of the EM iteration. Conditional probabilities \( z \) and the log-likelihood can be recovered from parameters via \texttt{estep}, while parameters can be recovered from conditional probabilities \( z \) using \texttt{mstep}:

\[
> \text{ms <- mstep( modelName = "VVV", data = irisMatrix, z = meVWiris$z)}
\]

\[
> \text{ms}
\]

\[
> \text{es <- estep( modelName = "VVV", data = irisMatrix, } \mu = \text{ms$mu, sigma = ms$sigma, pro = ms$pro)}
\]

\[
> \text{es}
\]
Figure 2: Uncertainty plot for the 3-cluster classification of Fisher's iris data via EM based on unconstrained Gaussian mixtures. The plot was created with `uncerPlot`, and shows the relative uncertainty of misclassified observations.

It is often useful in applications to invoke `estep` by an indirect or list call using the S-PLUS function `do.call`, in which the output of e.g. `mstep` is passed as a list without the need to explicitly identify individual parameters.

```r
> es <- do.call("estep", c(list(data = irisMatrix), ms))
> es
```

In Section 10.1, we show how to use `mstep` and `estep` for discriminant analysis.

### 5 Bayesian Information Criterion

`MCLUST` provides a function `bic` to compute the Bayesian Information Criterion (BIC) [19] given the maximized loglikelihood for model and the data dimensions. The BIC is the value of the maximized loglikelihood with a penalty for the number of parameters in the model, and allows comparison of models with differing parameterizations and/or differing numbers of clusters. In general the larger the value of the BIC, the stronger the evidence for the model and number of clusters (see, e.g. [11]). The following shows the BIC calculation in
MCLUST for the 3-cluster classification Fisher’s iris data with the varying variance spherical model:

```r
> hcEIIiris <- hc(modelName = "EII", data = irisMatrix)
> cl <- hclass(hcEIIiris, 3)

> emEst <- me(modelName = "VII", data = irisMatrix, z = unmap(cl))

> n <- nrow(irisMatrix)
> d <- ncol(irisMatrix)

> bic( modelName = "VII", loglik = emEst$loglik, n = n, d = d, G = 3)
[1] -853.8133

> do.call("bic", emEst)
[1] -853.8133
```

The next section describes functions that combine hierarchical clustering, EM, and BIC in a comprehensive model-based clustering strategy.

# 6 Cluster Analysis

MCLUST provides two functions, Mclust and EMclust, for cluster analysis combining hierarchical clustering, EM, and BIC. In both functions, hierarchical clustering is used to initialize EM for various parameterizations of the Gaussian model. Mclust is intended to be a simplified function for one-step model-based clustering, with reasonable defaults. EMclust has more options and more flexibility, although it may be more complicated to use.

## 6.1 Mclust

The input to Mclust is the data and the minimum and maximum numbers of groups to consider. Mclust compares BIC values for parameters optimized via EM for the models EII, VII, EEI, VVI, EEE, WVV. All models are initialized with the classification from hierarchical clustering based on the unconstrained VVV model. The output includes the parameters of maximum-BIC model, and the corresponding classification and uncertainty. The following is an example of the use of Mclust with Fisher’s iris data:

```r
> irisMclust <- Mclust(irisMatrix)

> summary(irisMclust)

  best model: ellipsoidal, unconstrained with 2 groups
```

An S-PLUS object produced by Mclust has the following components:
> names(irisMclust)
[1] "BIC" "bic" "classification" "uncertainty"
[5] "n" "d" "G" "z"
[9] "mu" "sigma" "pro" "loglik"
[13] "modelName"

Mclust has an associated plot method for displaying its results.

> plot(irisMclust,irisMatrix)

make a plot selection (0 to exit):

1: plot: BIC
2: plot: Pairs
3: plot: Classification (2-D projection)
4: plot: Uncertainty (2-D projection)
5: plot: All
Selection:

Figure 3 shows BIC, classification, and uncertainty plots corresponding to the model fit to the iris data.
Figure 3: Plots associated with the mclust function. Top: BIC. Lower Left: Projection of the optimal classification. Lower Right: Projection of the uncertainty of the optimal classification (see Section 9).
6.2 EMclust

EMclust has more flexibility than Mclust for clustering, by allowing choice of hierarchical clustering (which need not be model-based) as input to initialize EM and choice of models for EM. Users can obtain parameters and clustering results through summary for any model or number of mixture components specified, rather than just the maximum-BIC model as in Mclust.

The input to EMclust includes the data, a list of models to apply in the EM phase, the desired numbers of groups to consider. Default starting values for EM are obtained from applying the function hc for model-based hierarchical clustering to the data using the unconstrained model WWV.

EMclust returns the BIC values for all of the chosen models and number of clusters, together with auxiliary information that is used by the corresponding summary method for recovering parameter values. The following is an example of the use of EMclust with Fisher’s iris data:

\[ \text{irisBIC <- EMclust(irisMatrix)} \]

\[ \text{irisBIC} \]

**BIC:**

<table>
<thead>
<tr>
<th></th>
<th>EII</th>
<th>VII</th>
<th>EEI</th>
<th>VEI</th>
<th>EVI</th>
<th>VVI</th>
<th>EEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1804.0854</td>
<td>-1804.0854</td>
<td>-1527.1308</td>
<td>-1527.1308</td>
<td>-1527.1308</td>
<td>-1527.1308</td>
<td>-829.9782</td>
</tr>
<tr>
<td>2</td>
<td>-1123.4115</td>
<td>-1012.2352</td>
<td>-1047.9786</td>
<td>-961.2929</td>
<td>-1017.3295</td>
<td>-867.5728</td>
<td>-688.0972</td>
</tr>
<tr>
<td>3</td>
<td>-878.7652</td>
<td>-853.8133</td>
<td>-818.0635</td>
<td>-784.1658</td>
<td>-812.8672</td>
<td>-759.6687</td>
<td>-632.9658</td>
</tr>
<tr>
<td>4</td>
<td>-784.3098</td>
<td>-783.8263</td>
<td>-740.4955</td>
<td>-721.5350</td>
<td>-752.5491</td>
<td>-725.1132</td>
<td>-591.4097</td>
</tr>
<tr>
<td>6</td>
<td>-715.7147</td>
<td>-705.7813</td>
<td>-698.8104</td>
<td>-680.5938</td>
<td>-752.2155</td>
<td>-726.9666</td>
<td>-621.8183</td>
</tr>
<tr>
<td>7</td>
<td>-700.3690</td>
<td>-705.0659</td>
<td>-688.4205</td>
<td>-664.6910</td>
<td>-749.2664</td>
<td>-729.2289</td>
<td>-613.4585</td>
</tr>
<tr>
<td>8</td>
<td>-686.0964</td>
<td>-710.5799</td>
<td>-666.9473</td>
<td>-662.2667</td>
<td>-743.7443</td>
<td>-741.9637</td>
<td>-622.4215</td>
</tr>
<tr>
<td>9</td>
<td>-694.5239</td>
<td>-703.3490</td>
<td>-683.6092</td>
<td>-672.0079</td>
<td>-782.4065</td>
<td>-772.4925</td>
<td>-638.2063</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>EEV</th>
<th>VEV</th>
<th>VVV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-829.9782</td>
<td>-829.9782</td>
<td>-829.9782</td>
</tr>
<tr>
<td>2</td>
<td>-644.5997</td>
<td>-561.7285</td>
<td>-574.0178</td>
</tr>
<tr>
<td>3</td>
<td>-617.7022</td>
<td>-562.5519</td>
<td>-580.8400</td>
</tr>
<tr>
<td>4</td>
<td>-613.4447</td>
<td>-603.9274</td>
<td>-628.9642</td>
</tr>
<tr>
<td>5</td>
<td>-621.6908</td>
<td>-635.2096</td>
<td>-683.8194</td>
</tr>
<tr>
<td>6</td>
<td>-669.7087</td>
<td>-681.3057</td>
<td>-711.5716</td>
</tr>
<tr>
<td>7</td>
<td>-682.3451</td>
<td>-707.9405</td>
<td>-752.7982</td>
</tr>
<tr>
<td>8</td>
<td>-722.7178</td>
<td>-735.8582</td>
<td>-790.7023</td>
</tr>
<tr>
<td>9</td>
<td>-772.5045</td>
<td>-790.6323</td>
<td>-824.8824</td>
</tr>
</tbody>
</table>

\[ \text{plot(irisBIC)} \]

<table>
<thead>
<tr>
<th>EII</th>
<th>VII</th>
<th>EEI</th>
<th>VEI</th>
<th>EVI</th>
<th>VVI</th>
<th>EEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A&quot;</td>
<td>&quot;B&quot;</td>
<td>&quot;C&quot;</td>
<td>&quot;D&quot;</td>
<td>&quot;E&quot;</td>
<td>&quot;F&quot;</td>
<td>&quot;G&quot;</td>
</tr>
<tr>
<td>&quot;H&quot;</td>
<td>&quot;I&quot;</td>
<td>&quot;J&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14
The BIC values for this example are shown in Figure 4, with the key to symbols returned by the plot method. Application of the summary function to this result gives a classification as well as model parameters (not printed):

```r
> irisSummary <- summary(irisBIC, irisMatrix)
> irisSummary

classification table:
    1  2
  50 100

best BIC values:
VEV,2  VEV,3  VVV,2
-561.7285 -562.5519 -574.0178

best model: ellipsoidal, equal shape
```

The best model among those fitted by EMclust is the equal-shape model VEV, with 2 clusters. The same model with 3 clusters has a BIC value that is little different from the maximum; the conclusion is that there are either 2 or 3 clusters in the data under these models. The Fisher's iris data consists of length and width measurements for the petals and sepals of 50 replicates in each of three species (see e.g. help(iris)). The 2 cluster EMclust result separates one species from the other while the 3 cluster result nearly separates the three species (there are 5 misclassifications out of 150).

Optimal parameter and z values are available through the summary function associated with EMclust objects, which has arguments allowing the summarizing information to be restricted to a subset of the number of clusters and models. The summary function requires the data to be supplied as an argument in addition to the output from EMclust. The model giving the best classification (according to the BIC) is recovered from summary by default, and the summary object includes the classification as a component along with the corresponding model parameters and z values. The next best classification is the 3 group classification with the equal-shape model VEV. Parameters associated with this classification can be recovered via summary as follows:

```r
> iris3 <- summary(irisBIC, irisMatrix, G = 3, modelNames = "VEV")
> iris3

classification table:
    1  2  3
  50 45 55

best BIC value:
VEV,3
-562.5519
```
Figure 4: BIC values from EMclust for the models A - EII, B - VII, C - EEI, D - VEI, E - EVI, F - VVI, G - EEE, H - EEV, I - VEV, J - VVV applied to Fisher’s iris data.

best model: ellipsoidal, equal shape

For a complete analysis, it may be desirable to try varying models, initialization strategies for EM and values for the convergence tolerance, as well as using permutations or subsets of the observations, and/or perturbations the data, to see if the classification remains stable. Scaling or otherwise transforming the data may also affect the results. It is advisable to examine the data beforehand, in case (for example) the dimensions can be reduced due to highly correlated variables.

Finally, it is important to take into account numerical issues in cluster analysis. The EM computations break down when the covariance corresponding to one or more components becomes ill-conditioned (singular or nearly singular). In general they cannot proceed if clusters contain only a few observations or if the observations they contain are very nearly colinear. If EM for a model having a certain number of components is applied to a mixture in which there are actually fewer groups, then it may fail due to ill-conditioning. For more information about ill-conditioned matrices and the resulting consequences for computation, see [14]. The EM functions in MCLUST compute and monitor the conditioning of the covariances, and an error condition is issued when the associated covariance appears to be nearly singular, as determined by a threshold with the default value Mclust$\$eps.
6.3 Clustering with Noise and Outliers

MCLUST uses mixture model which has a single term representing noise as a first order Poisson process to handle noisy data:

$$\prod_{i=1}^{n} \left[ \frac{\tau_0}{V} + \sum_{k=1}^{K} \tau_k \phi_k(x_i | \theta_k) \right],$$

in which $V$ is the hypervolume of the data region, and $\tau_k \geq 0; \sum_{k=0}^{G} \tau_k = 1$. This model has been used successfully in a number of applications [2, 7, 4, 5].

The basic model-based clustering method needs to be modified when the data contains noise. First, a good initial noise estimate must be obtained. Some possible methods for denoising include a Voronoi method [1] and a nearest-neighbor method [3]. Next, hierarchical clustering is applied to the denoised data. Finally, EM based on the Gaussian model with the added noise term (2) is applied to the entire data set, with the data removed in the denoising process as the initial noise estimate.

MCLUST provides a function EMclustN for model-based clustering with noise. In the following example, Poisson noise is added to Fisher's iris data, with a random initial estimate for the noise.

```r
irisMatrix <- matrix(aperm(iris, c(1,3,2)), 150, 4)
dimnames(irisMatrix) <- list(NULL, dimnames(iris)[[2]])

b <- apply( irisMatrix, 2, range)
n <- 450
set.seed(0)
poissonNoise <- apply(b, 2, function(x, n=n)
                      runif(n, min = min(x)-.1, max = max(x)+.1), n = n)
set.seed(0)
noiseInit <- sample(c(T,F), size=150+450, replace=T, prob=c(3,1))
Bic <- EMclustN(data=rbind(irisMatrix, poissonNoise), noise = noiseInit)
```

The data and classification are show in Figure 5.
Figure 5: Cluster analysis of Fisher’s iris data with added Poisson noise. Upper Left: A projection Fisher’s iris data (circles) with 450 Poisson noise points (small dots). Upper Right: EMclustN classification. Lower: BIC from EMclustN.
7 Simulation from Mixture Densities

Because the cluster analysis strategy described in Section 6 is a model-fitting procedure, it has uses other than grouping observations. Given the parameters for a mixture model, data can be simulated from that model for evaluation and verification. The function `sim` allows simulation from mixture models generated by MCLUST functions. Besides the model, `sim` allows a seed as input for reproducibility. As an example, below we simulate from the 2 and 3 class equal-shape models produced by EMclust in Section 6.2:

```r
> model2 <- summary(irisBIC, irisMatrix, G=2, modelName="VEV")
> sim2 <- do.call("sim", c(list(n=1000, seed=0), model2))

> model3 <- summary(irisBIC, irisMatrix, G=3, modelName="VEV")
> sim3 <- do.call("sim", c(list(n=1000, seed=0), model3))

> par(pty = "s") # square plotting surface

> coordProj(data=sim2, mu = model2$mu, sigma = model2$sigma)

> coordProj(data=sim3, mu = model3$mu, sigma = model3$sigma)
```

A projection of each of these simulations is shown in Figure 6, along with the same projection of Fisher’s iris data.
Figure 6: A projection of data simulated from EMclust models for Fisher’s iris data. Upper Left: 1000 data points simulated from the 2-component VEV model. Upper Right: 1000 data points simulated from the 3-component VEV model. Lower: Fisher’s iris data showing classification.
8 Density Estimation

The clustering capabilities of \texttt{MCLUST} can also be viewed as a general strategy for multivariate density estimation.

As an example, we focus on density estimation for the location of maple trees in the Lansing Woods [13, 15] (see Figure 7).

![Lansing Woods Maple Trees](image)

\textbf{Figure 7: Lansing Woods maple trees.}

\texttt{MCLUST} can be used for density estimation as follows: First, use \texttt{EMclust} to get a model for the Lansing Woods maples:

\begin{verbatim}
> maples <- lansing[, "species"] == "maple", -3]
> maplesBIC <- EMclust(maples)
> maplesModel <- summary(maplesBIC, maples)
\end{verbatim}

Next, use \texttt{dens} to get the density of a given point relative to that model. The following computes the density on a $100 \times 100$ grid. The function \texttt{grid1} forms a one dimensional grid of a given size over a given range of values, while \texttt{grid2} forms a two dimensional grid given two sequences of values.
> x <- grid1(150, range = c(0,1))
> y <- grid1(100, range = c(0,1))
> xy <- grid2(x,y)
> xyDens <- do.call("dens", c(list(data = xy), maplesModel))

The result can be plotted using S-PLUS functions contour, persp, or image.

> par(pty = "s")
> Z <- matrix(log(xyDens), nrow = length(x), ncol = length(y))
> contour(x = x, y = y, z = Z, nlevels = 20)
> persp(x = x, y = y, z = Z)
> image(x = x, y = y, z = Z)

Figure 8 shows the density estimate for both the default convergence tolerance \(Mclust$\text{tol} = 1.e-5\) and a relaxed convergence tolerance \(1.e-4\) in \textsc{EMclust}. For information on creating classification and density displays for model-based clustering results, see Section 9.

Probably the most common application for density estimation is discriminant analysis, for which a detailed discussion is given in Section 10.

9 Displays

Once parameters values are available, projections of the data showing the means and standard deviations of the corresponding clusters may be plotted. In the two-dimensional case, density and uncertainty surfaces may also be plotted.

9.1 Plotting Two-Dimensional Results

The function \texttt{mclust2Dplot} may be used for displaying the classification, uncertainty, or classification errors for \texttt{MCLUST} models of two-dimensional data. In the following example, classification and uncertainty plots are produced for the Lansing Woods maples example of Section 8. We have defined a function \texttt{plotMaples1} that calls \texttt{mclust2Dplot} to facilitate plotting the results already obtained for Fisher's iris data.

> plotMaples1
function(type)
{
  out <- do.call("mclust2Dplot", c(list(data = maples, type = type),
                               maplesModel))

  invisible()
}

> par(pty = "s", mfrow = c(1,2)) ## plotting parameters

> plotMaples1(type = "classification")
> title("Classification", cex = 0.75)
> plotMaples1(type = "uncertainty")
> title("Uncertainty", cex = 0.75)

The resulting plots are displayed in Figure 9.

The function `surfacePlot` may be used for displaying the density or uncertainty for `mclust` models of two-dimensional data. It also returns the grid coordinates and corresponding surface values. The following example shows how to display density and uncertainty surfaces for the Lansing Woods maples models.

> plotMaples2
function(type, what, transformation)
{
  out <- do.call("surfacePlot", c(maplesModel, 
    list(data = maples, type = type, what = what, transformation = transformation))
  invisible()
}

> par(pty = "s", mfrow = c(3,2)) ## plotting parameters

> plotMaples2(type = "contour", what = "density", transformation = "log")
> plotMaples2(type = "contour", what = "uncertainty", transformation = "log")

> plotMaples2(type = "persp", what = "density", transformation = "log")
> plotMaples2(type = "persp", what = "uncertainty", transformation = "log")

> plotMaples2(type = "image", what = "density", transformation = "log")
> plotMaples2(type = "image", what = "uncertainty", transformation = "log")

The resulting plots are displayed in Figure 10. Note that a gray scale or color scheme may need to be specified on the display device in order to view the image plots.
9.2 Plotting Multidimensional Results

9.2.1 Coordinate Projections

To plot coordinate projections in MCLUST, use the function `coordProj`. In what follows we have defined a function `coordIris` that calls `coordProj` to facilitate plotting the results already obtained for the iris data. Note that `truth` is used only for the errors plot. If the plot type isn’t specified, `coordProj` will offer a menu of options if there is more than one possibility.

```r
> coordIris
function(dimens, type)
{
  out <- do.call("coordProj", c(list(data = irisMatrix, truth = irisClass,
                                 dimens = dimens, type = type), meVWViris))
  invisible()
}
```

```r
> par(pty = "s", mfrow = c(2,3))
> coordIris( dimens = c(1,2), type = "classification")
> title("3 Group Classification, Unconstrained Model")
> coordIris( dimens = c(1,2), type = "uncertainty")
> title("Uncertainty")
> coordIris( dimens = c(1,2), type = "errors")
> title("Classification Errors")
```

Plots for `dimens = c(1,2)` and `dimens = c(3,4)` are displayed in Figure 11.
Figure 8: Density estimation for the Lansing Woods maples. Left: relaxed convergence tolerance \( \text{tol} = 1.e-4 \). Right: default convergence tolerance \( \text{tol} = 1.e-5 \). First Row: BIC from model-based clustering. Second Row: \texttt{EMclust} classification with circles/ellipses indicating the standard deviation of each component. Third Row: Density contours with the location of the maples superimposed.
Figure 9: Classification (left) and uncertainty (right) for the \texttt{EMclust} modeling of the Lansing Woods maples created with \texttt{mclust2Dplot}. The circles shown are the standard deviations of each mixture component. In the classification plot, points in different classes are indicated by different symbols. In the uncertainty plot, the symbols have the following meaning: large filled symbols, 95\% quantile of uncertainty; smaller open symbols, 75–95\% quantile; small dots, first three quartiles of uncertainty.
Figure 10: Density (left column) and uncertainty (right column) surfaces for the Lansing Woods maples. Contour, perspective, and image plots are displayed in the first, second, and third rows respectively. A logarithmic scale was used for the contour plots.
Figure 11: Coordinate projections of Fisher’s iris data created with `coordProj`. Plots show the 3-cluster classification with associated uncertainty and classification errors for the unconstrained Gaussian mixture model (WVW).
9.2.2 Random Projections

To plot random projections in MCLUST, use the function randProj. In what follows we have defined a function randIris that calls randProj to facilitate plotting the results already obtained for Fisher’s iris data.

```r
> randIris
function(seed, type)
{
  out <- do.call("randProj", c(list(data = irisMatrix, truth = irisClass,
                                 seed = seed, type = type), meVWViris))
  invisible()
}

> par(pty = "s", mfrow = c(2,3))

> randIris( seed = 646, type = "classification")
> title("3 Group Classification, Unconstrained Model")

> randIris( seed = 646, type = "uncertainty")
> title("Uncertainty")

> randIris( seed = 646, type = "errors")
> title("Classification Errors")

Plots for seed = 646 and seed = 666 are displayed in Figure 12.
Figure 12: Random projections of Fisher's iris data created with `randProj`. Plots show the 3-cluster classification with associated uncertainty and classification errors for the unconstrained Gaussian mixture model (WWW).
10 Discriminant Analysis

In discriminant analysis, observations of known classification are used to classify others. **MCLUST** provides a number of functions that can be used for discriminant analysis. We demonstrate some possible methods applied to the Lansing Woods data [13, 15], which gives the spatial location of maple and hickory trees (see Figure 13).

![Lansing Woods Hickory and Maple Trees](image)

Figure 13: Location of hickory and maple trees in the Lansing Woods. There are 703 hickories (triangles) and 514 maples (squares) in the data set.
10.1 Discriminant Analysis using mstep and estep

MCLUST functions mstep and estep implementing the individual steps of the EM algorithm for Gaussian mixtures can be used for discriminant analysis. The idea is to produce a density estimate for the training data which is a mixture model, in which each known class is modeled by a single Gaussian term.

First, the parameterization giving the best model fit to the training data must be chosen. Most commonly, this would be done by leave-one-out cross validation. Leaving out one training observation at a time, MCLUST function cv1EMtrain fits each model using mstep, then classifies the observation that was left out using estep. The output of cv1EMtrain is the error rate for each model; that is, the fraction of left-out observations correctly classified by the model fit to the remaining observations.

Using the odd numbered observations in the Lansing Woods data as a training set, the result is:

```r
> odd <- seq(from=1, to=nrow(lansing), by=2)
> round(cv1EMtrain(data = lansing[odd,-3], labels = lansing[odd,3]),3)
   EII VII EEI VEI EVI VVI EEE EVV VVV
0.351 0.355 0.345 0.353 0.34 0.358 0.345 0.333 0.35 0.35
```

The equal shape, equal volume, varying orientation model EEV would be selected as the best model, although the error rates do not vary much among the models. When there are two training classes, the EEE model corresponds to linear discriminant analysis, while the VVV model corresponds to quadratic discriminant analysis (e.g. [16]).

To classify the even data points, we first compute the parameters corresponding to the EEV model for the odd data points using mstep, then use estep to get conditional probabilities z and a classification:

```r
> eeModd <- mstepEEV(data=lansing[odd,-3], z=unmap(lansing[odd,3]))
> eeZ <- do.call("estepEEV", c(eeModd, list(data=lansing[,,-3])))$z
> classError(map(eeZ[odd,]), lansing[odd,3])
[1] 0.3316913
```

```r
> even <- seq(from=2, to=nrow(lansing), by=2)
> classError(map(eeZ[even,]), lansing[even,3])
[1] 0.3125
```

The error rate for the training [odd-numbered] data is 33%, while for the test [even-numbered] data it is 31%. The results for this analysis are displayed in the left hand plot of Figure 14.

Another option for model selection that is quicker than cross validation to compute is to select the best fitting model via BIC after using mstep to fit each model to the training data. A function bicEMtrain is provided in MCLUST for this purpose. For the Lansing Woods data, BIC for the models fitted to the odd-numbered observations is:

```r
> round(bicEMtrain(lansing[odd,-3], labels = lansing[odd,3]),1)
   EII VII EEI VEI EVI VVI EEE EVV VVV
-512.1 -526.5 -520 -533.7 -539.5 -554.7 -519.7 -534.3 -549.7 -553.4
```
Figure 14: Discriminant analysis using `mstep` and `estep`. The plots show the ellipses corresponding to the standard deviation of each of the two Gaussians used to fit the training data. Filled symbols represent misclassified training [odd-numbered] and test [even-numbered] observations. Left: Model EIV selected by leave-one-out crossvalidation. Right: Model EII selected via BIC.

According to BIC, the spherical model with equal volume EII is the best model. The error rate for the training data is about 35%, while for the test data it is 33%:

```
> eiiModd <- mstepEII(data=lansing[odd,-3], z=unmap(lansing[odd,3]))
> eiiZ <- do.call("mstepEII", c(eiiModd, list(data=lansing[, -3])))$z
> classError(map(eiiZ[odd,]), lansing[odd,3])
[1] 0.3513957
```

```
> even <- seq(from = 2, to = nrow(lansing), by = 2)
> classError(map(eiiZ[even,]), lansing[even,3])
[1] 0.3305921
```

The results for this analysis are displayed in the right hand plot of Figure 14. Although the error rate for the test data is somewhat higher for BIC than for crossvalidation, it should be noted that it took more than 10 minutes to execute `cv1EMtrain` on this training data, while `bicEMtrain` executed in about half a second.\(^4\)

### 10.2 Mixture Discriminant Analysis via MclustDA

In Section 10.1, discriminant analysis was accomplished modeling the training data by a mixture density with a single Gaussian component for each class. That section also showed how to choose the appropriate cross-cluster constraints to give the lowest training error rate using either leave-one-out crossvalidation or BIC. Better discriminant analysis results can often be obtained by using model-based clustering to fit a Gaussian mixture model as a density estimate for each class in the training set.

\(^4\)This is not a precise algorithmic timing comparison because crossvalidation can be accomplished more efficiently for these models using updating schemes, and because the interpreted language S-PLUS is used for parts of the training code.
10.2.1  mclustDA

If both training and test sets are given in advance, the simplified function mclustDA can be used for discriminant analysis. Its input is the training data and associated class labels, and the test data. The output of mclustDA includes the mixture models for the training data, the classification of both the test data and training data under the model, posterior probabilities for the test data, and the training error rate.

```r
> discrim <- mclustDA(trainingData = lansing[odd,-3], labels = lansing[odd,3],
                      testData = lansing[even,-3])
> discrim

    trainClass mclustModel numGroups
    hickory     hickory     EEV    7
    maple      maple      VII    7

training error rate: 0.204

> classError(discrim$testClassification, lansing[even,3])
[1] 0.2351974
```

The error rates for mclustDA classification of the Lansing Woods data are about 20% and 24% for the training [odd-numbered] and test [even-numbered] data, respectively.

The associated plot method offers a selection of coordinate projection plots.

```r
> par(pty = "s", mfrow = c(2,2)) # plotting parameters
> plot(discrim, testData = lansing[odd,-3], labels = lansing[odd,3],
      testData = lansing[even,-3], identify = T)
```

make a plot selection (0 to exit):

1: plot: Training and Test Data
2: plot: Training Data - known classification
3: plot: Test Data - mclustDA classification
4: plot: Training Data - misclassified observations
5: plot: All
Selection: 5

Figure 15 shows the result of selecting All. The training models are shown in Figure 16.

10.2.2  mclustDAtest and mclustDAtest

Often more flexibility is required in discriminant analysis. For example, a suitable training set may need to be chosen and/or it may be desirable to test additional data after a training density has already been established. Since training takes much more time than testing, it can be advantageous to separate training and testing computations. In contrast to mclustDA,
Figure 15: Plots associated with mclustDA. Upper Left: the training [odd-numbered/circles] and test [even-numbered/crosses] Lansing Woods data. Upper Right: the training data with known classification. Lower Left: the mclustDA classification of the data. Lower Right: the errors (filled symbols) in using the mclustDA model to classify the training data.
mclustDA also allows users to choose training model parameterizations, and selects from among all available models as a default. The output of mclustDAtrain is a list, each element being the model for each class.

In the simplest case, a single Gaussian could be fit to each training class. This is similar to the discriminant analysis procedure of Section 10.1, except that in MclustDA a model in for each class of the training data is chosen separately, instead of choosing a parameterized mixture model (which may have cross-cluster constraints) for the training data. In discriminant analysis via M-step and E-step (Section 10.1) restricted to the unconstrained (VIV) model, the maximum likelihood Gaussian is fitted to each component. MclustDA uses BIC, which adds a penalty term to the maximized log-likelihood that increases with the number of parameters, to select the model.

By default, mclustDAtrain will fit up to nine components for each possible model. Results for the odd-numbered observations in the Lansing Woods data are as follows:

```r
> train <- mclustDAtrain(data = lansing[odd,-3], labels = lansing[odd,3])
   EEV    VII
        7    7

> summary(train)
hickory model: EEV
classification table:
   1 2 3 4 5 6 7
31 57 89 31 65 29 50

maple model: VII
classification table:
   1 2 3 4 5 6 7
```
mclustDAtrain chooses the 7-class EEV model for the hickory training class and the 7-class VII model for the maple training class.

The density of observations under the training models can be obtained using mclustDAtest, while the classification and posterior probabilities of the test data can be recovered from the summary function for mclustDAtest:

```r
> testTrain <- mclustDAtest(models = train, data = lansing[odd, -3])

> names(summary(testTrain))
[1] "classification" "z"

> classError(summary(testTrain)$classification, lansing[odd, 3])
[1] 0.2036125

> testTest <- mclustDAtest(models = train, data = lansing[even, -3])

> classError(summary(testTest)$classification, lansing[even, 3])
[1] 0.2351974
```
11 One Dimensional Data

The MCLUST functions for clustering, density estimation and discriminant analysis can be applied to one-dimensional as well as multidimensional data. Analysis is somewhat simplified since there are only two possible models — equal variance (denoted E) or varying variance (denoted V).

11.1 Clustering

As an example, we use simulated data consisting of two clusters with variance 1 centered at -9 and 9, respectively, and one cluster with variance 4 centered at 0:

```r
> set.seed(999)
> x <- c(rnorm(300, -9), rnorm(400, 0, sd = 2), rnorm(300, 9))
```

Cluster analysis for one-dimensional data can be carried out as for two and higher dimensions, except that there is a special plotting function mclust1Dplot:

```r
> par(mfrow = c(2,2))
> xBIC <- EMclust(x)
> plot(xBIC)
E  V
"1" "2"
```

```r
> xModel <- summary(xBIC, x)
> do.call("mclust1Dplot", c(list(data = x), xModel))
mclust1Dplot: make a plot selection (0 to exit):
1: classification
2: uncertainty
3: density
4: all
Selection:
```

Figure 17 shows the BIC, classification, uncertainty, and density for this simulated example.

11.2 Discriminant Analysis

To illustrate discriminant analysis on one-dimensional data, we use the simulated data set x from the previous section as a training set, with the group centered at the origin as one class and the remaining groups as another.

```r
> set.seed(999)
> x <- c(rnorm(300, -9), rnorm(400, 0, sd = 2), rnorm(300, 9))
> xClass <- c(rep(1,300),rep(2,400),rep(1,300))
```
Figure 17: Model-based clustering of one-dimensional data. Clockwise from upper left: BIC, classification, uncertainty, and density from EMclust applied to the simulated one-dimensional example. In the classification plot, all of the data is displayed at the bottom, with the separated classes shown different levels above.

We use the following simulated data as a test set:

```r
> set.seed(0)
> y <- c(rnorm(100, -9), rnorm(100, 0, sd = 2), rnorm(100, 9))
> yClass <- c(rep(1,100),rep(2,100),rep(1,100))
```

Discriminant analysis via EM (Section 10.1) is possible in one dimension. Both leave-one-out crossvalidation and BIC choose the equal variance model E in the training stage:
> round(cv1EMtrain(x, labels=xClass), 3)
  E  V
  0.4 0.057
> round(bicEMtrain(x, labels=xClass), 3)
  E  V
  -6785.284  -7046.888

The varying variance model V is chosen by cross-validation, while the equal-variance model is chosen by BIC. The training and test errors for the data with these models are as follows.

Equal variance:

> xEmstep <- mstep(modelName = "E", data = x, z = unmap(xClass))

> xEz <- do.call("estep", c(list(data = x), xEmstep))$z
> classError(map(xEz), xClass)  ## training error
[1] 0.4
Warning messages:
  no assignment to 2 in: map(xEz)
> yEz <- do.call("estep", c(list(data = y), xEmstep))$z
> classError(map(yEz), yClass)  ## test error
[1] 0.3333333
Warning messages:
  no assignment to 2 in: map(yEz)

Varying variance:

> xVmstep <- mstep(modelName = "V", data = x, z = unmap(xClass))

> xVz <- do.call("estep", c(list(data = x), xVmstep))$z
> classError(map(xVz), xClass)  ## training error
[1] 0.057

> yVz <- do.call("estep", c(list(data = y), xVmstep))$z
> classError(map(yVz), yClass)  ## test error
[1] 0.0433333

For discriminant analysis via MclustDA (Section 10.2):

> xtrain <- mclustDAtrain(x, labels = xClass)
  E  X
  2 1
> xTest <- summary(mclustDAtest(x, xtrain))
> classError(xTest$classification, xClass)  ## training error
[1] 0

> yTest <- summary(mclustDAtest(y, xtrain))
> classError(yTest$classification, yClass)  ## testing error
[1] 0.003333333

40
11.3 "mclust" option for S-PLUS density function

MCLUST includes an augmented version of the S-PLUS function density for computing the density of a one-dimensional data set. A method argument has been added, with the option to specify method = "mclust" to have the density computed via model-based clustering, instead of the default kernel estimate.

> xdens.default <- density(x, n = 100)
> xdens.mclust <- density(x, n = 100, method = "mclust")

The resulting densities for the simulated data used in the previous two sections are displayed in Figure 18. A simulation study [18] showed that mixtures of normals with equal variance can give substantially better density estimates than kernel-based methods for one-dimensional data.

Figure 18: A comparison of method = "mclust" and default kernel options for the augmented S-PLUS density function on the one-dimensional simulated example. Upper Left: The underlying density (solid line) and the computed density with default kernel method (dotted line). Upper Right: The underlying density (solid line) and the computed density with method = "mclust" (dotted line). Lower: Errors for method = "mclust" (solid line) and the default (dotted line).
12 Extensions

12.1 Large Data Sets

EMclust includes a provision for using a subsample of the data in the hierarchical clustering phase before applying EM to the full data set. This strategy is often adequate for large data sets, although it may miss small groups. Iterative methods for handling such cases are discussed in section 10.3 of [11]. The following example uses a random sample of size 100 in the initial hierarchical clustering phase of EMclust applied to the iris data:

```r
> n <- nrow(irisMatrix)
> S <- sample(1:n, size = 100)
> irisBIC <- EMclust(irisMatrix, subset = S)
```

For very large data sets, the discrimination capability of MCLUST can be used for classification. First, cluster analysis with the methodology of EMclust can be performed on a subset of the data. Then the remaining data points can then be classified (in reasonable sized blocks) using one of the discriminant analysis techniques described in section 10.

12.2 High Dimensional Data

Models in which the orientation is allowed to vary between clusters (EEV, VEV, EVV, VVV), have $O(d^2)$ parameters per cluster, where $d$ is the dimension of the data. For this reason, MCLUST may not work well or may otherwise be inefficient for these models when applied to high-dimensional data. It may still be possible to analyze such data with MCLUST by restriction to models with fewer parameters (e.g. spherical or diagonal models), or else by applying a dimension-reduction technique such as principal components.

Some of the more parsimonious models (e.g. spherical, diagonal, or fixed covariance) can be applied to datasets in which the number of observations is smaller than the data dimension.
13 Function Summary

13.1 Hierarchical Clustering

hc Merge sequences for model-based hierarchical clustering.
klass Classifications corresponding to hc results.

13.2 Parameterized Gaussian Mixture Models

eem EM algorithm (starting with E-step).
em Me EM algorithm (starting with M-step).
estep E-step of the EM algorithm.
mstep M-step of the EM algorithm.
mvn One-component fit.

13.3 Density Computation for Parameterized Gaussian Mixtures

cdens Component density (without mixing proportions).
dens Mixture density.

13.4 Model-based Clustering / Density Estimation

EMclust BIC computation; clusters and models through summary.
Mclust Combines EMclust and its summary (fewer options).
density S-PLUS one-dimensional density function with method = "mclust" option.

13.5 Discriminant Analysis

Class Densities as Mixture Components

cv1EMtrain Training via leave-one-out crossvalidation.
bicEMtrain Training via BIC.
estep E-step of the EM algorithm.
mstep M-step of the EM algorithm.

Parameterized Gaussian Mixture for Class Densities (MclustDA)

mclustDAtrain MclustDA training.
mclustDAtest MclustDA density; classification via summary.
mclustDA Combines mclustDAtrain and mclustDAtest (fewer options).
13.6 Support for Modeling and Classification

```r
.Mclust  vector of default values.
mclustOptions set MCLUST options.
  map   Convert conditional probabilities to a classification.
  unmap Convert a classification to indicator variables.
  bic   BIC for parameterized Gaussian mixture models.
  sim   Simulate data from a parameterized Gaussian mixture model.
compareClass Compare two classifications.
mapClass Mapping between two classifications.
classError Classification error.
classErrors Identification of classification errors.
sigma2decomp Convert mixture covariances to decomposition form.
decomp2sigma Convert decomposition form to mixture covariances.
  grid1 One-dimensional grid.
  grid2 Two-dimensional grid.
```

13.7 Plotting Functions

13.7.1 One-Dimensional Data

```r
mclust1Dplot Classification, uncertainty, density and/or classification errors.
```

13.7.2 Two-Dimensional Data

```r
mclust2Dplot Classification, uncertainty, and/or classification errors.
surfacePlot Contour, image, or perspective plot of either density or uncertainty.
```

13.7.3 More than Two Dimensions

Classification, uncertainty, and/or classification errors.

```r
  coordProj coordinate projections
  randProj random projections
  spinProj random projection followed by reflection or rotation
```

13.7.4 Other Plotting Functions

```r
clPairs pairs plot showing classifications
uncerPlot relative uncertainty of misclassified observations
plot.Mclust plots associated with Mclust results
plot.mclustDA plots associated with mclustDA results
```
References


45


