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

MCLUST 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. MCLUST 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 and N00014-96-1-0330, and by National Institutes of Health grant 1 R01 CA94212-01

[†]Thanks go to Ron Wehrens for comments and suggestions leading to improvements in the MCLUST documentation and software.

[1]Insightful Corp., Seattle, WA USA — `http://www.insightful.com/splus`

[2]A version of MCLUST is also available with an R language interface instead of S-PLUS (ported by Ron Wehrens `http://www-cac.sci.kun.nl/people/rwehrens`). R is freely available for noncommercial use — see `http://lib.stat.cmu.edu/R/CRAN`.

Contents

List of Tables

List of Figures

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

$$\phi_k(\mathbf{x} \mid \mu_k, \Sigma_k) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_k)^T \Sigma_k^{-1} (\mathbf{x}_i - \mu_k)\right\}, \tag{1}$$

where $\mathbf{x}$ represents the data, and k is an integer subscript specifying a particular cluster. Clusters are ellipsoidal, centered at the means μ_k. The covariances Σ_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 Σ_k, and λ_k is a scalar. The orientation of the principal components of Σ_k is determined by D_k, while A_k determines the shape of the density contours; λ_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, 7]. 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 Σ_k currently available in `MCLUST` for hierarchical clustering (HC) and/or EM for multidimensional data. ('•' in the appropriate column indicates availability).

identifier	Model	HC	EM	Distribution	Volume	Shape	Orientation
`EII`	λI	•	•	Spherical	equal	equal	NA
`VII`	$\lambda_k I$	•	•	Spherical	variable	equal	NA
`EEI`	λA		•	Diagonal	equal	equal	coordinate axes
`VEI`	$\lambda_k A$		•	Diagonal	variable	equal	coordinate axes
`EVI`	λA_k		•	Diagonal	equal	variable	coordinate axes
`VVI`	$\lambda_k A_k$		•	Diagonal	variable	variable	coordinate axes
`EEE`	λDAD^T	•	•	Ellipsoidal	equal	equal	equal
`VVV`	$\lambda_k D_k A_k D_k^T$	•	•	Ellipsoidal	variable	variable	variable
`EEV`	$\lambda D_k A D_k^T$		•	Ellipsoidal	equal	equal	variable
`VEV`	$\lambda_k D_k A D_k^T$		•	Ellipsoidal	variable	equal	variable

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:

```
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

```
> dyn.open(".../MCLUST/S.so")
> 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, unzip this file to obtain 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. As an example of the use of `hc` and `hclass`, consider Fisher's iris data [9], 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)
```

[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`'.

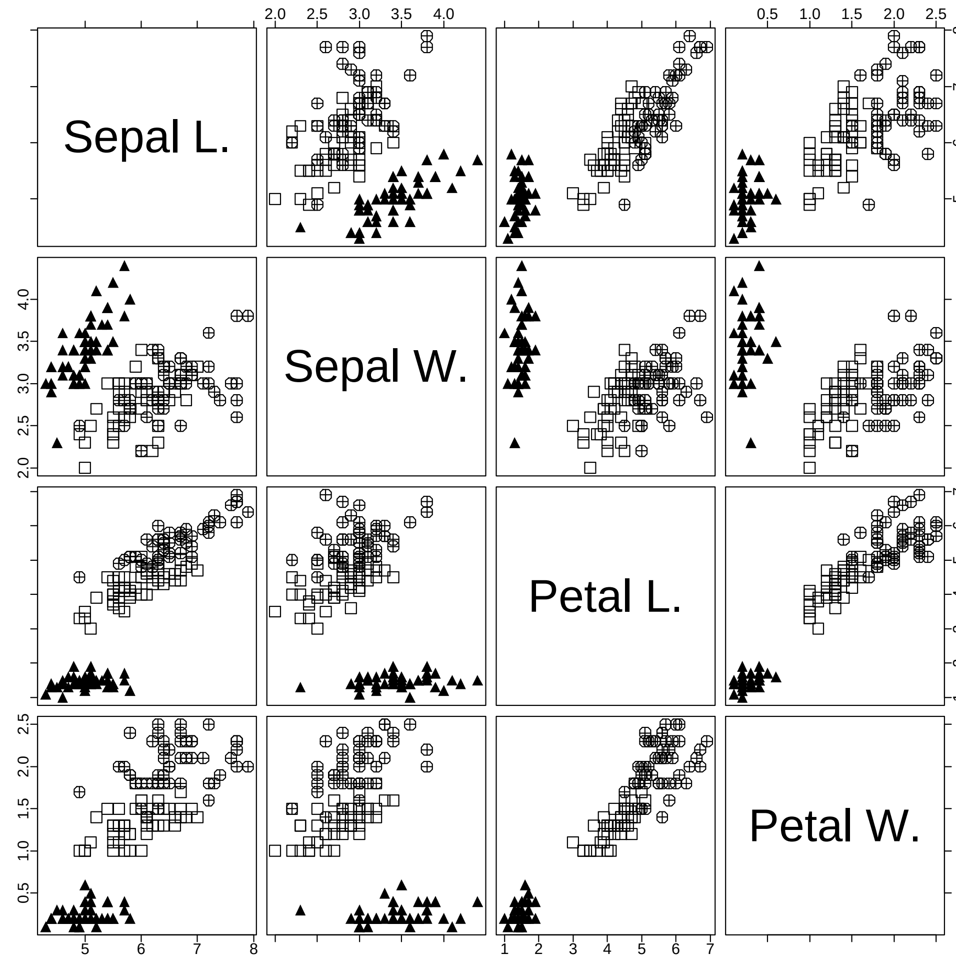


Figure 1: Pairs plot of Fisher's iris data showing classifcation into species.

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:

```
> 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

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

and the sums of the columns of z converge to n times the mixing proportions τ_k, where n is the number of observations in the data. Here G is the number of groups in the data, which is fixed in the EM algorithm. The parameterizations of Σ_k currently available for EM in MCLUST are listed in Table 1. They are a subset of the parameterizations discussed in [7], 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:

```
> hcVVViris <- hc( modelName = "VVV", data = irisMatrix) # unconstrained model
> cl <- hclass(hcVVViris, 3)                  # 3-group hc classification
> meVVViris <- 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 coverting 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:

```
> clPairs( irisMatrix, map(meVVViris$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", "VVE", "VEV").

`eps` A tolerance for terminating iterations due to ill-conditioning, such as near singularity in covariance matrices. The default is `.Mclust$eps` which is the relative machine precision `2.220446e-16` in the release.

Although these are in some sense hidden by the defaults, they may have a significant affect on results and should be taken into consideration in analysis. An example in which `tol = 1.e-4` and `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:

```
> uncer <- 1 - apply( meVVViris$z, 1, max)
```

The S-PLUS function `quantile` applied to the uncertainty gives a measure of the quality of the classification.

```
> quantile(uncer)
   0%  25%          50%         75%      100%
    0    0 1.949809e-08 0.001384177 0.3378195
```

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 `uncerPlot`, as is done below for the `iris` example (see Figure 2):

```
> irisClass <- rep(dimnames(iris)[[3]], rep(50,3))

> uncerPlot(z = meVVViris$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

`MCLUST` includes functions `estep` and `mstep` implementing the individual steps of the EM iteration. Conditional probabilities z and the log likelihood can be recovered from parameters via `estep`, while parameters can be recovered from conditional probabilities z using `mstep`:

```
> ms <- mstep( modelName = "VVV", data = irisMatrix, z = meVVViris$z)
> ms

> es <- estep( modelName = "VVV", data = irisMatrix,
               mu = ms$mu, sigma = ms$sigma, pro = ms$pro)
> es
```

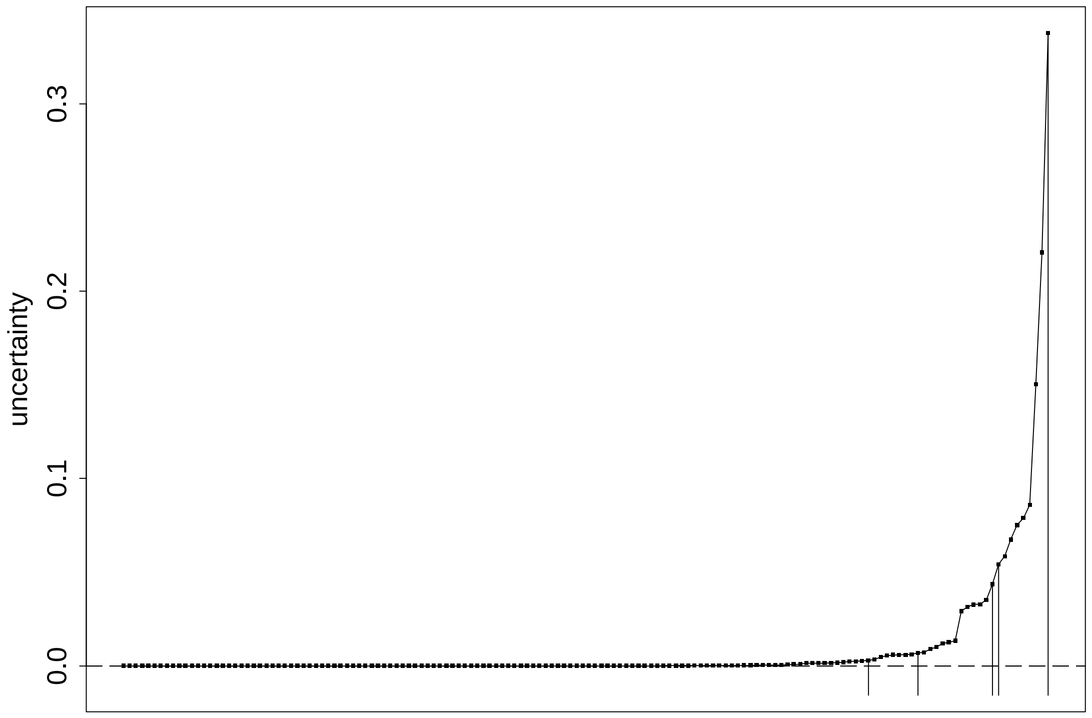


Figure 2: Uncertainty plot for the the 3-cluster classification of Fisher's iris data via EM based on unconstrained Gaussian mixtures. The plot was created with `uncerPlot`, and shows the relative uncertainty of misclassified observations.

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.

```
> 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:

```
> 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, VVV. 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:

```
> 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.

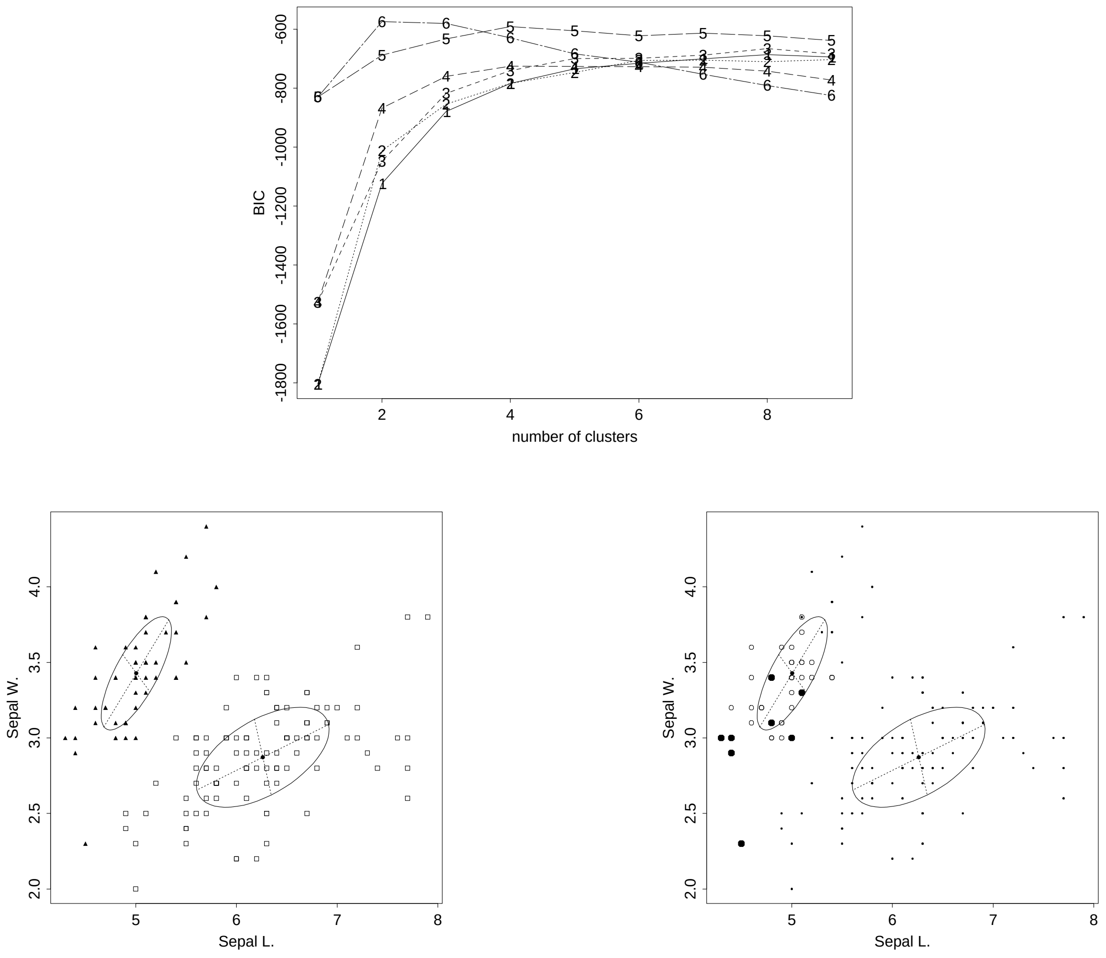


Figure 3: Plots associated with the `Mclust` function. Top: BIC. Lower Left: Projection of the optimal classfication. Lower Right: Projection of the uncertainty of the optimal classification (see Section 9).

6.2 EMclust

`EMclust` has more flexibilty 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` is the data, a list of models to apply in the EM phase, the desired numbers of groups to consider, and as hierarchical clustering tree in the same format as produced by the function `hc` for model-based hierachical clustering (the default is to apply `hc` for the unconstrained model `VVV` to the data). It 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:

```
> irisBIC <- EMclust(irisMatrix)

> irisBIC

 BIC:
        EII        VII        EEI        VEI        EVI        VVI       EEE
1 -1804.0854 -1804.0854 -1527.1308 -1527.1308 -1527.1308 -1527.1308 -829.9782
2 -1123.4115 -1012.2352 -1047.9786  -961.2929 -1017.3295  -867.5728 -688.0972
3  -878.7652  -853.8133  -818.0635  -784.1658  -812.8672  -759.6687 -632.9658
4  -784.3098  -783.8263  -740.4955  -721.5350  -752.5491  -725.1132 -591.4097
5  -734.3863  -746.9928  -699.4019  -708.0611  -720.7265  -725.9635 -604.9287
6  -715.7147  -705.7813  -698.8104  -680.5938  -752.2155  -726.9666 -621.8183
7  -700.3690  -705.0659  -688.4205  -664.6910  -749.2664  -729.2289 -613.4585
8  -686.0964  -710.5799  -666.0947  -662.2667  -743.7443  -741.9637 -622.4215
9  -694.5239  -703.3490  -683.6092  -672.0079  -782.4065  -772.4925 -638.2063

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

> plot(irisBIC)
 EII VII EEI VEI EVI VVI EEE EEV VEV VVV
 "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

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):

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

> irisSummary
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2

 uncertainty (quartiles):
   0%  25%  50%         75%         100%
    0    0    0 8.91745e-12 0.0002025613

 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 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 best classification (according to the BIC) is recovered from `summary` by default. The next best classification is the 3 group classification with the unconstrained model `VVV`. Parameters associated with this classification can be recovered via `summary` as follows:

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

> iris3
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 2
 [75] 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3

 uncertainty (quartiles):
   0%  25%          50%          75%        100%
```

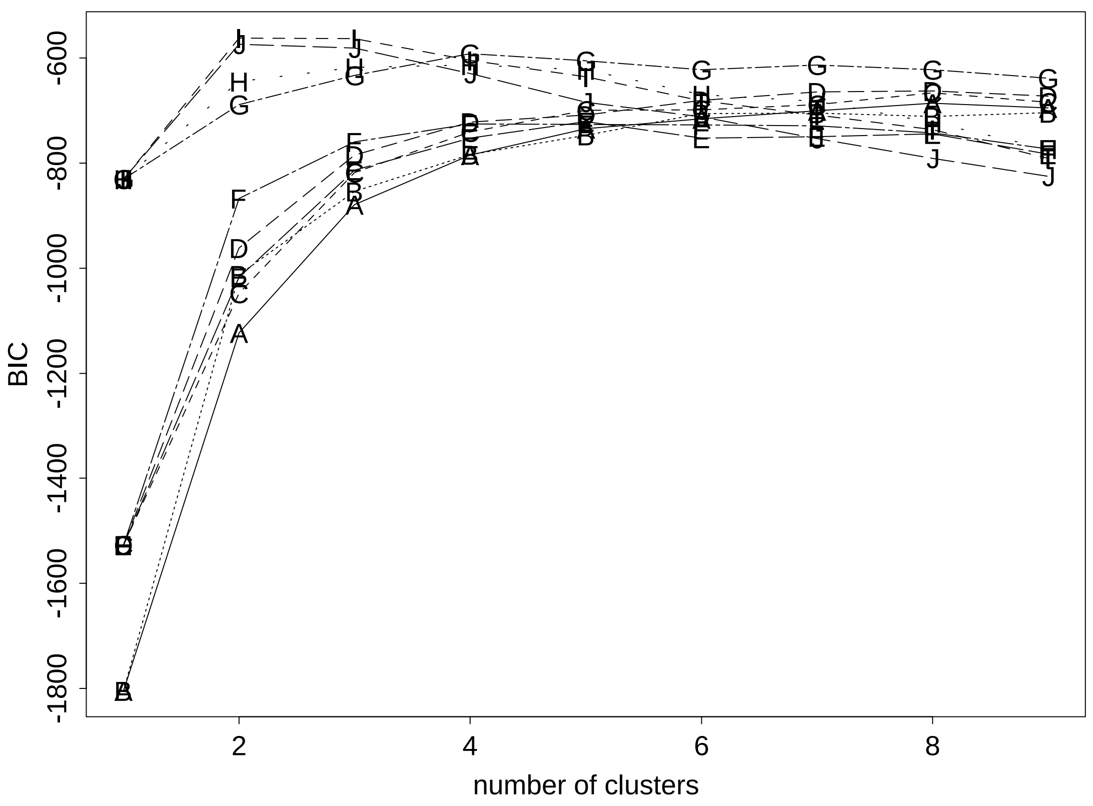


Figure 4: BIC values from `EMclust` for the models A - `EII`, B - `VII`, C - `EEI`, D - `VEI`, E - `EVI`, F - `VVI`, G - `EEE`, H - `EEV`, I - `VEV`, J - `VVV` applied to Fisher's iris data.

```
    0     0 1.048137e-06 0.002528291 0.4058713

 best BIC value:
     VEV,3
 -562.5519

 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:

$$\prod_{i=1}^{n} \left[\frac{\tau_0}{V} + \sum_{k=1}^{K} \tau_k \phi_k(\mathbf{x}_i \mid \theta_k) \right], \tag{2}$$

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, 8, 5, 6].

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 Voronoï method [1] and a nearest-neighbor method [4]. 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.

```
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 = x[1]-0.1, max = x[2]+.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.

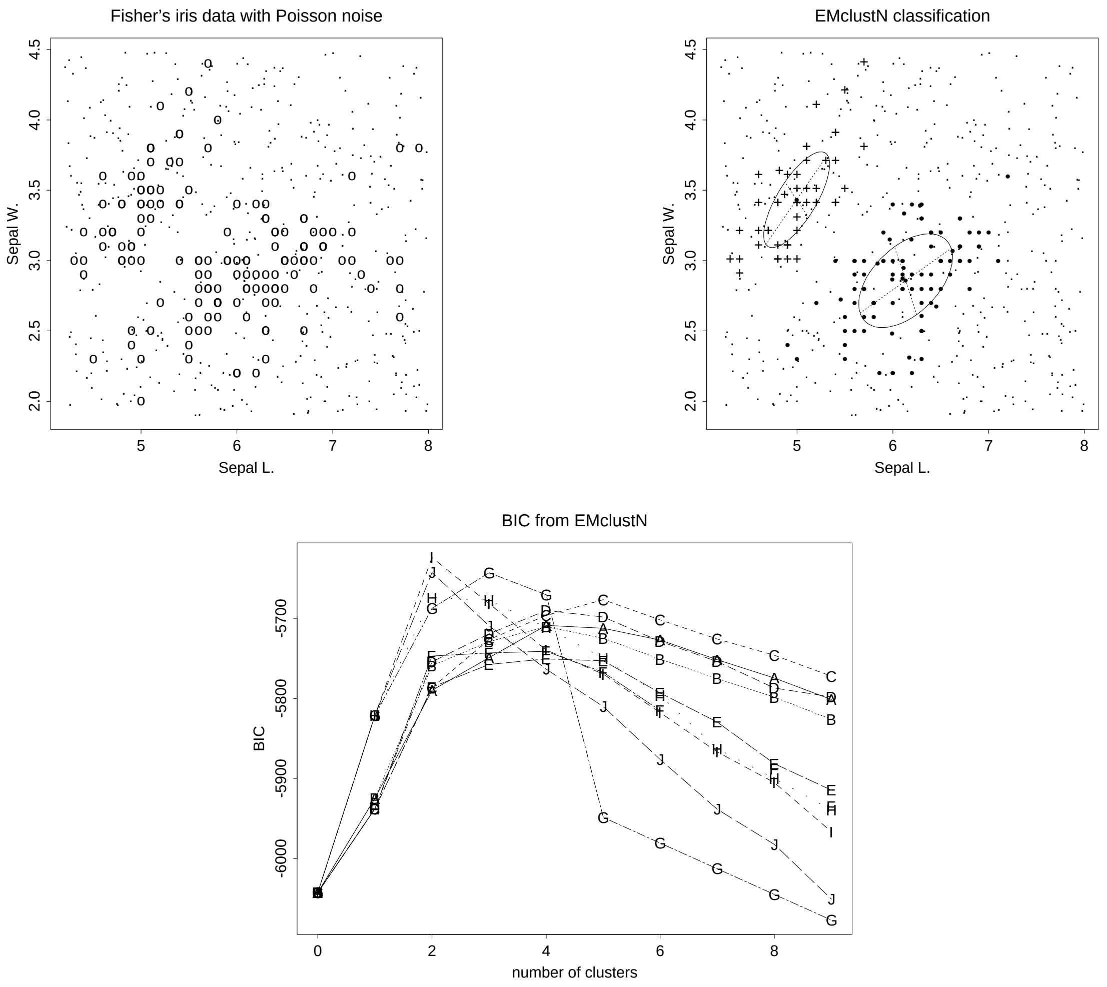


Figure 5: Cluster analysis of Fisher's iris data with added Poisson noise. Upper Left: A projection Fisher's iris data (circles) with 450 Poisson noise points (small dots). Upper Right: `EMclustN` classification. Lower: BIC from `EMclustN`.

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:

```
> 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.

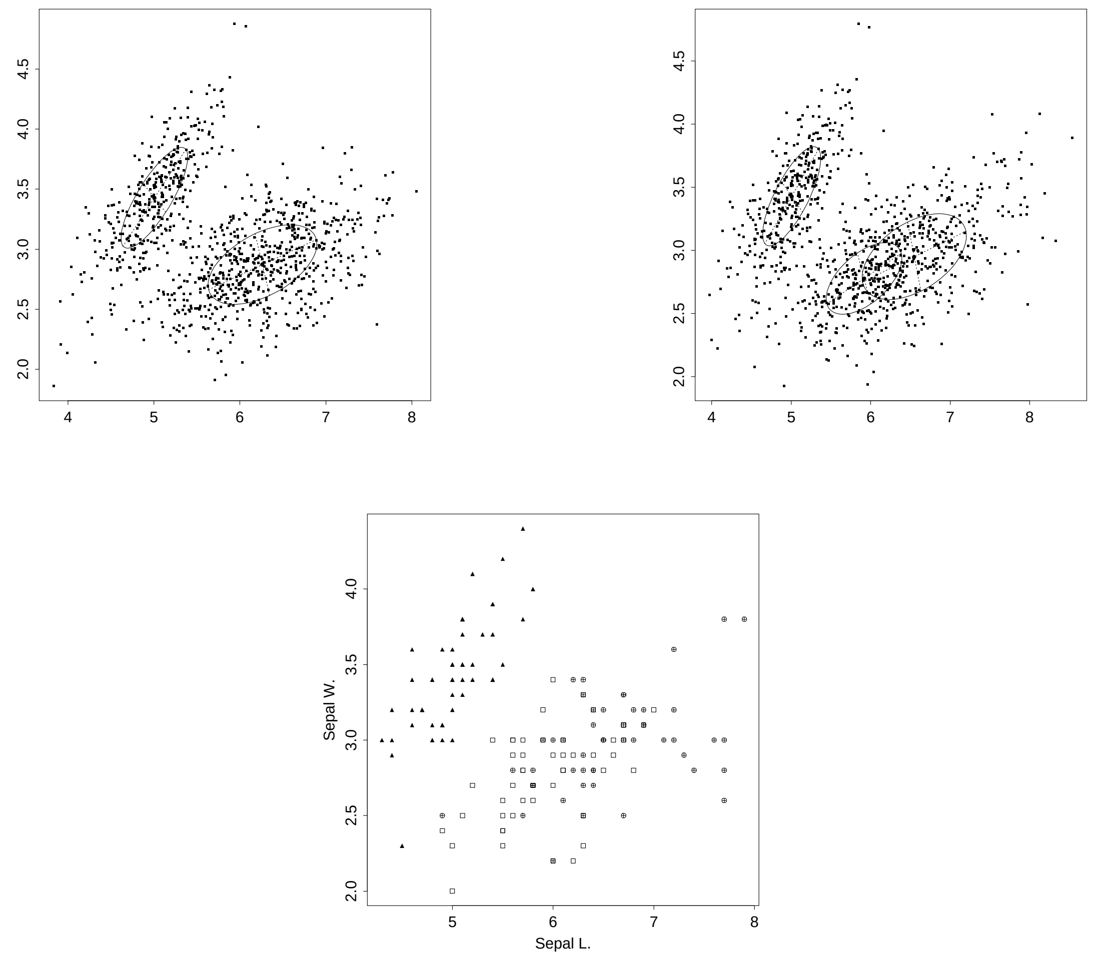


Figure 6: A projection of data simulated from `EMclust` models for Fisher's iris data. Upper Left: 1000 data points simulated from the 2-component `VEV` model. Upper Right: 1000 data points simulated from the 3-component `VEV` model. Lower: Fisher's iris data showing classfication.

8 Density Estimation

The clustering capabilities of 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).

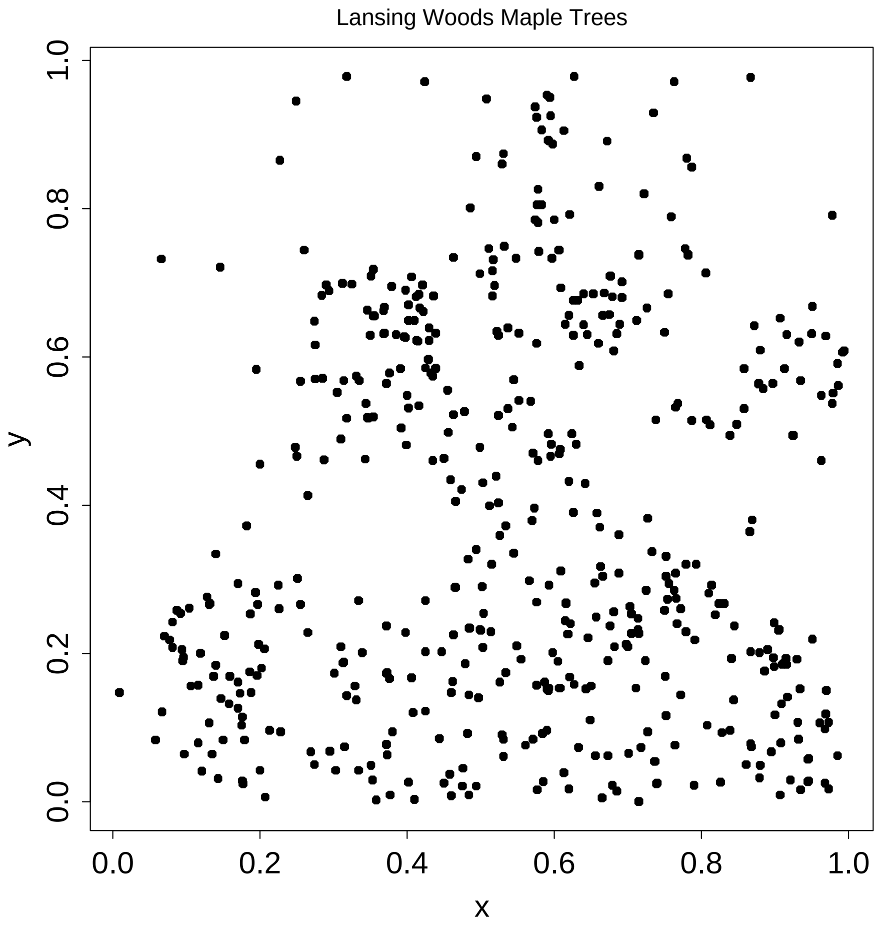


Figure 7: Lansing Woods maple trees.

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

```
> maples <- lansing[as.character(lansing[,"species"]) == "maple", -3]
> maplesBIC <- EMclust(maples)
> maplesModel <- summary(maplesBIC,maples)
```

Next, use dens to get the density of a given point relative to that model. The following computes the density on a 100×100 grid. The function grid1 forms a one dimensional grid of a given size over a given range of values, while 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$tol = 1.e-5` and a relaxed convergence tolerance 1.e-4 in `EMclust`. For information on creating classification and density displays for model-based clustering results, see Section 9.

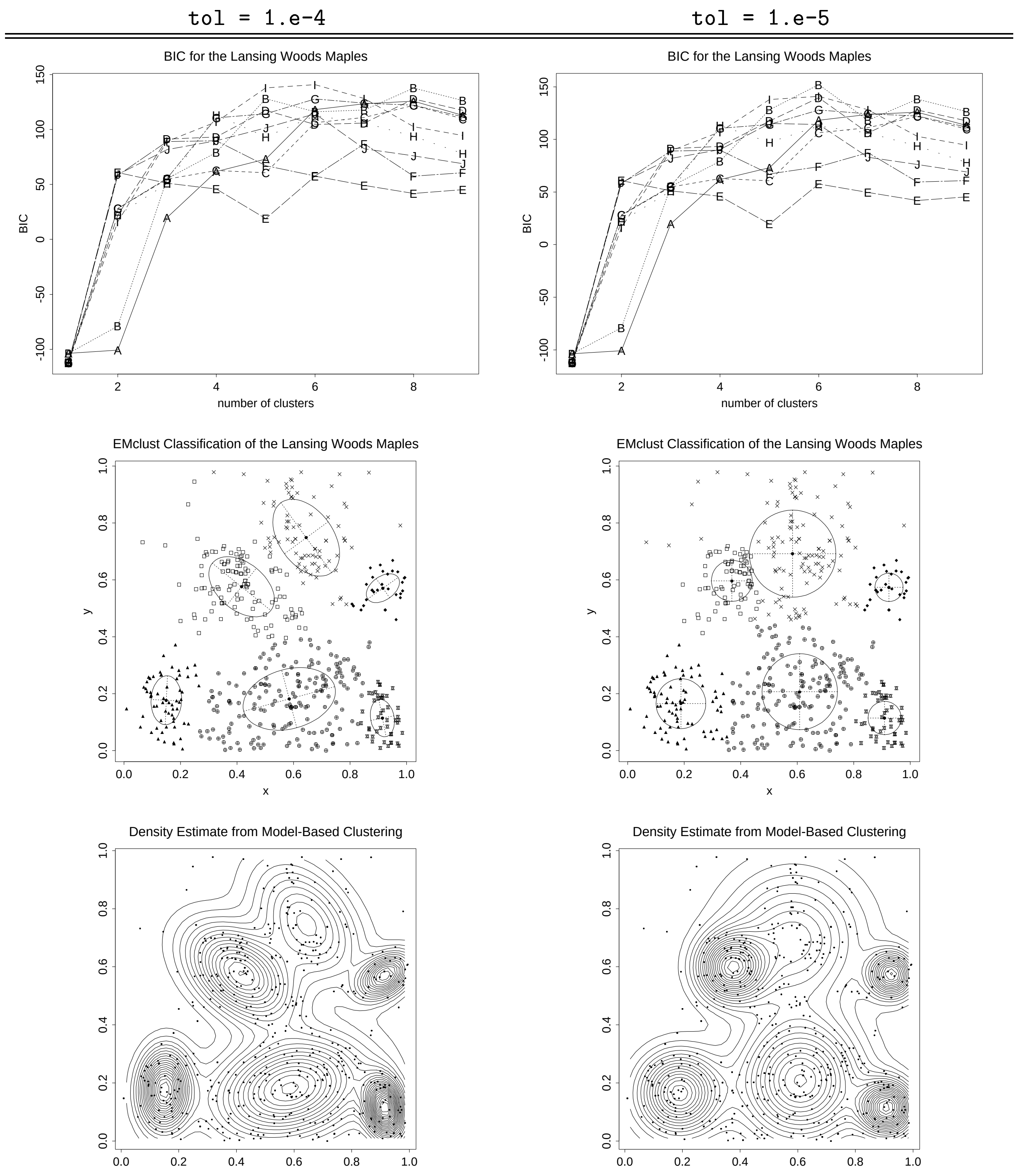
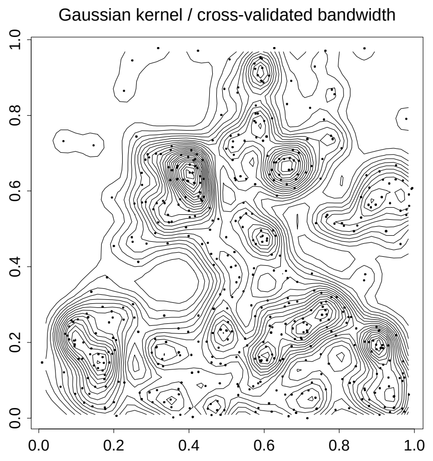


Figure 8: Density estimation for the Lansing Woods maples. Left: relaxed convergence tolerance `tol = 1.e-4`. Right: default convergence tolerance `tol = 1.e-5`. First Row: BIC from model-based clustering. Second Row: `EMclust` classification with circles/ellipses indicating the standard deviation of each component. Third Row: Density contours with the location of the maples superimposed.

Standard Gaussian kernel density estimates for this data are shown in Figure 9. The kernel densities and bandwidths were computed using the sm software (Bowman and Azzalini 1997).



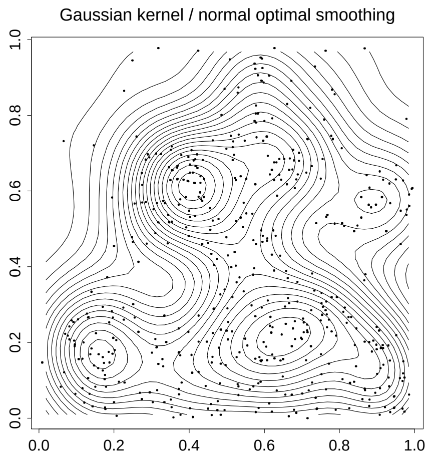


Figure 9: Gaussian kernel density estimates for the Lansing Woods maples. Left: Cross-validated bandwidth. Right: Normal optimal bandwidth.

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 mclust2Dplot may be used for displaying the classification, uncertainty, or classification errors for 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 plotMaples1 that calls 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()
}
```

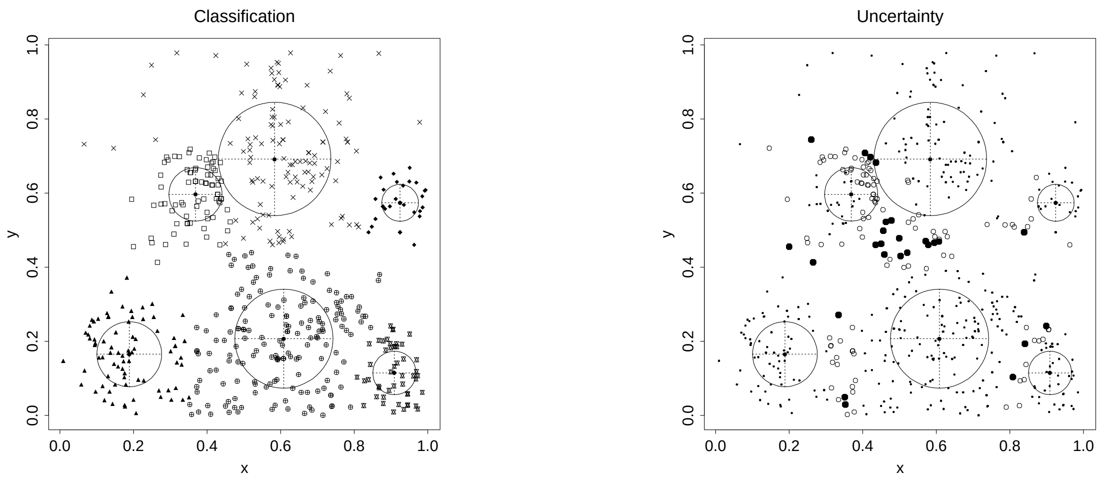


Figure 10: Classification (left) and uncertainty (right) for the `EMclust` modeling of the Lansing Woods maples created with `mclust2Dplot`. The circles shown are the standard deviations of each mixture component. In the classification plot, points in different classes are indicated by different symbols. 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.

```
> 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 10.

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 11. Note that a gray scale or color scheme may need to be specified on the display device in order to view the image plots.

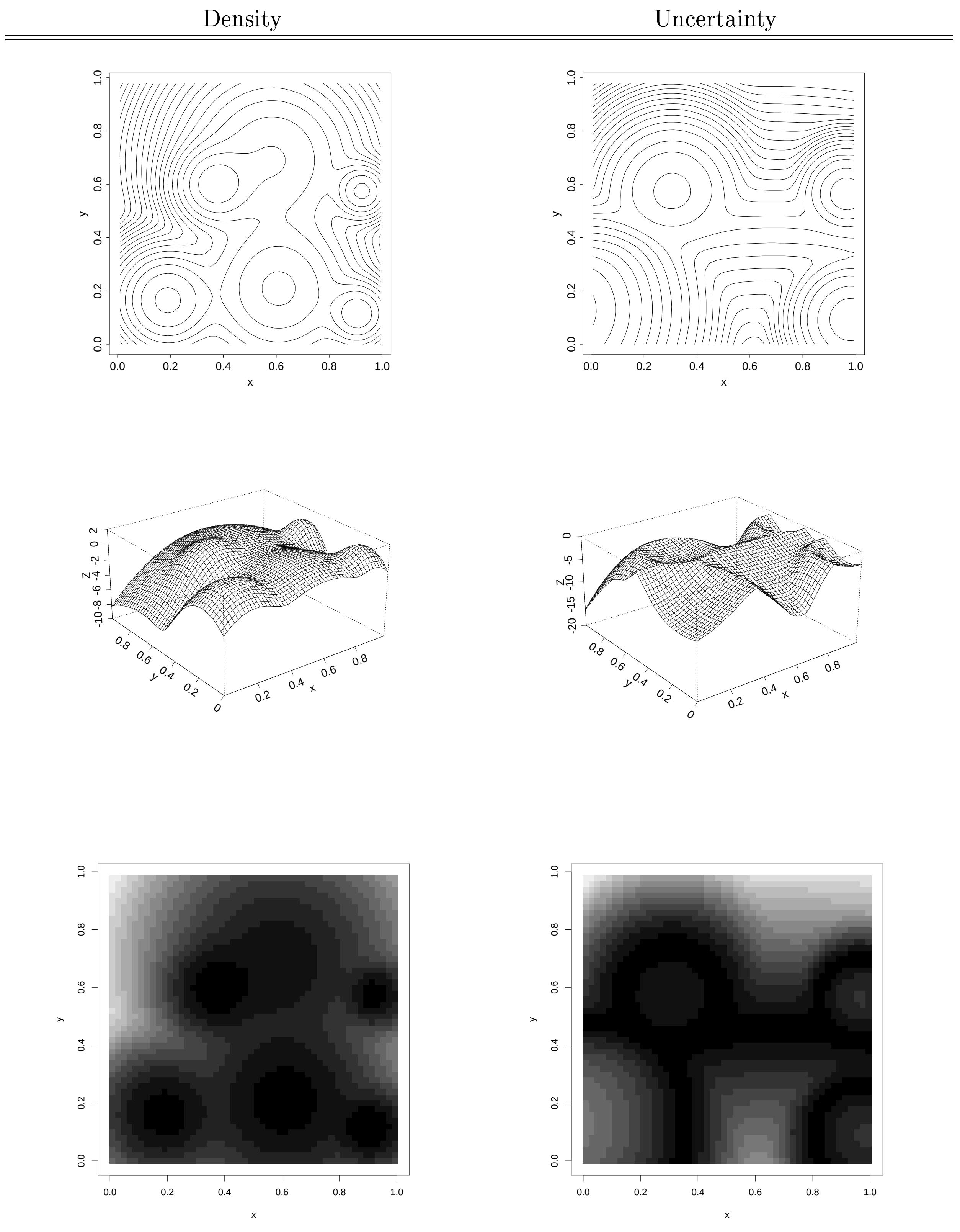


Figure 11: Density (left column) and uncertainty (right column) surfaces for the Lansing Woods maples. Contour, perspective, and image plots are displayed in the first, second, and third rows respectively.

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.

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

> 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 12.

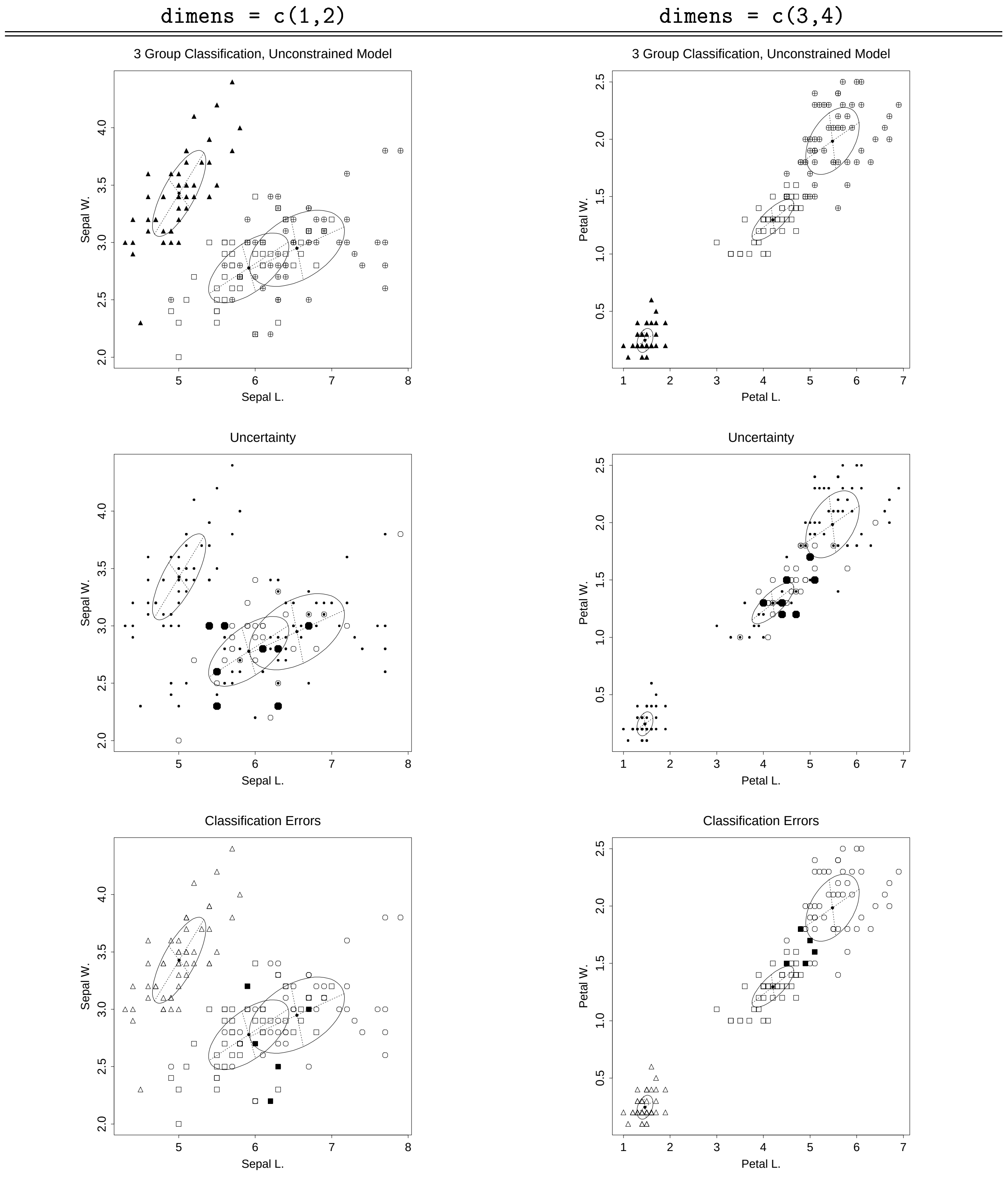


Figure 12: Coordinate projections of Fisher's iris data created with `coordProj`. Plots show the 3-cluster classification with associated uncertainty and classification errors for the unconstrained Gaussian mixture model (`VVV`).

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.

```
> randIris
function(seed, type)
{
 out <- do.call("randProj", c(list(data = irisMatrix, truth = irisClass,
                                   seed = seed, type = type), meVVViris))
 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 13.

`seed = 646` `seed = 666`

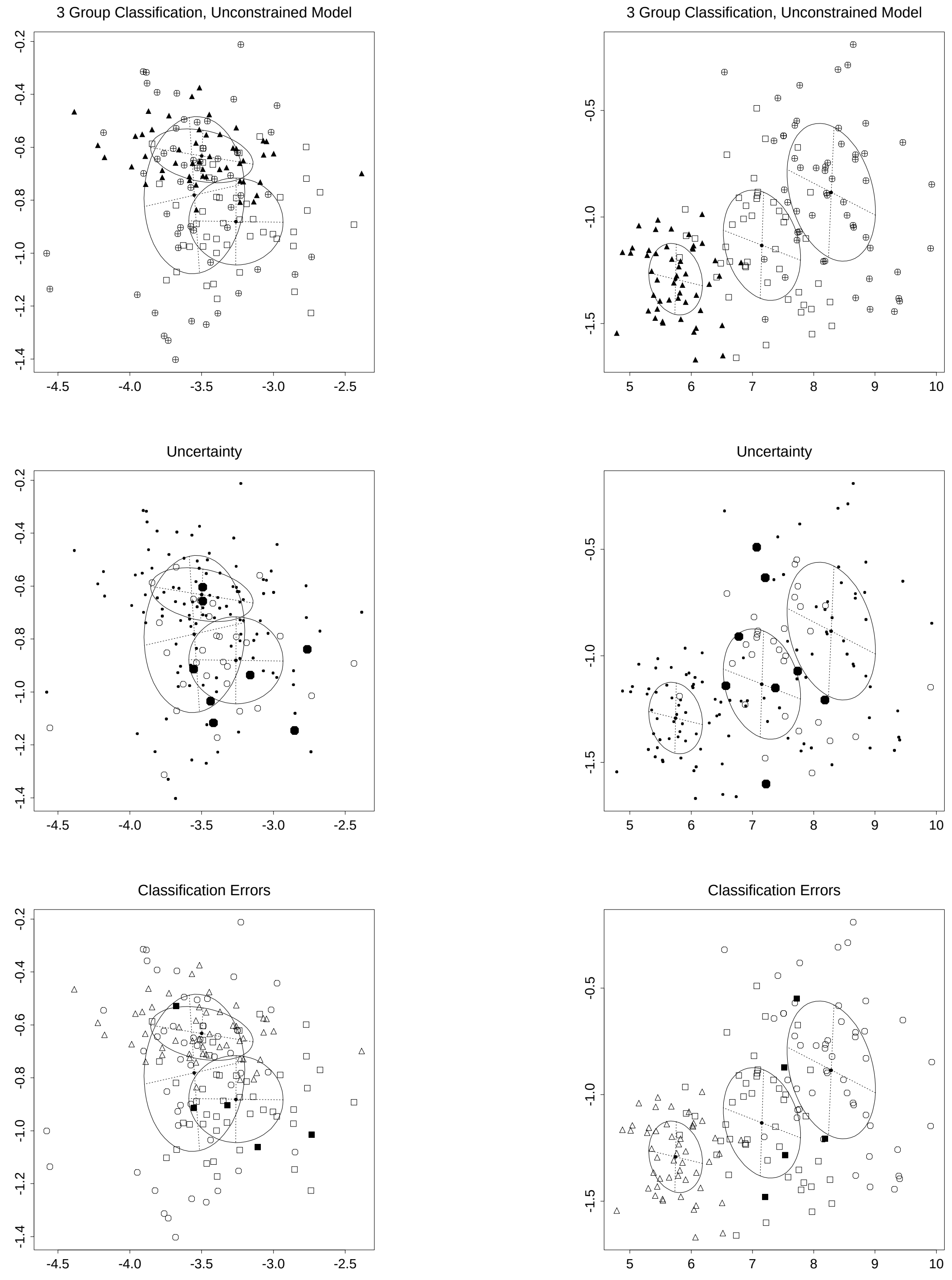


Figure 13: Random projections of Fisher's iris data created with `randProj`. Plots show the 3-cluster classification with associated uncertainty and classification errors for the unconstrained Gaussian mixture model (`VVV`).

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 14).

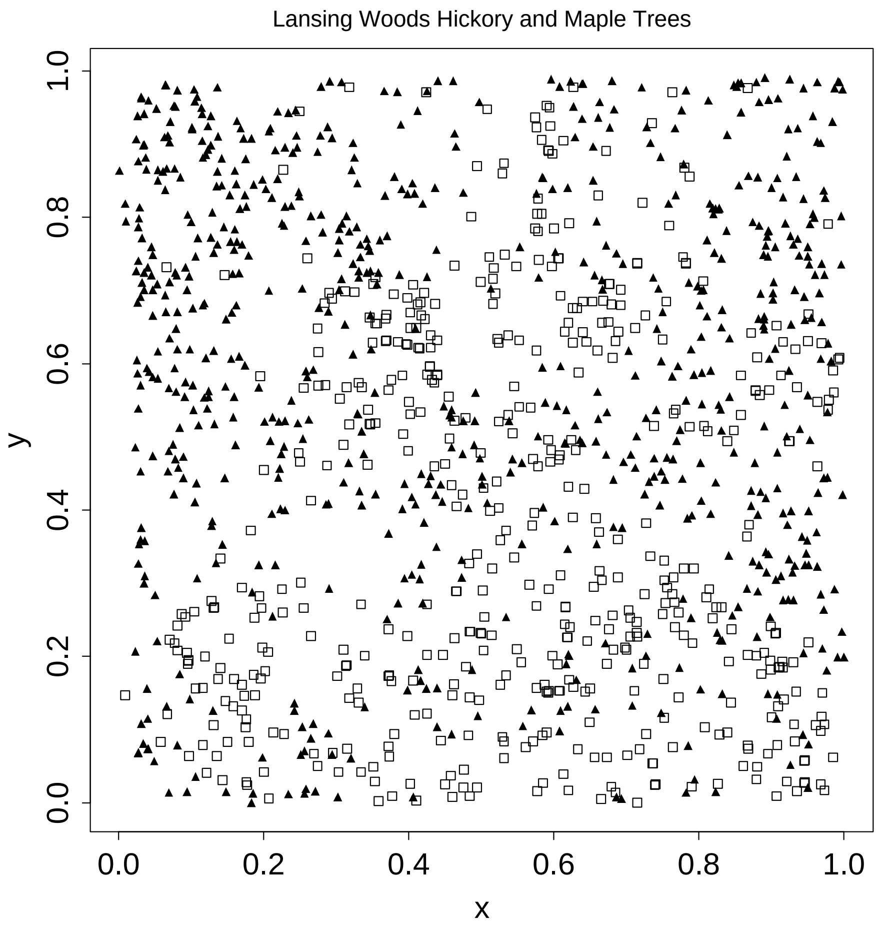


Figure 14: Location of Hickory and Maple Trees in the Lansing Woods. There are 703 hickories (triangles) and 514 maples (squares) in the data set.

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:

```
> 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   EEV  VEV  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. The `EEE` model corresponds to linear discriminant analysis, while the `VVV` model corresponds to quadratic discriminant analysis (e.g. [16]).

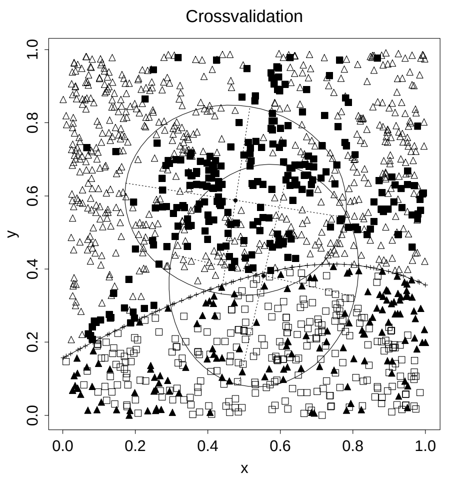
To classify the even data points, we first compute the parameters corresponding to the `EEV` model for the odd data points using `mstep`, then use `estep` to get conditional probabilities z and a classification:

```
> cv1Modd <- mstepEEV(data=lansing[odd,-3], z=unmap(lansing[odd,3]))
> cv1Zodd <- do.call("estepEEV", c(cv1Modd, list(data=lansing[odd,-3])))$z
> compClass(map(cv1Zodd), lansing[odd,3])$error
[1] 0.3316913

> even <- seq(from=2, to=nrow(lansing), by=2)
> cv1Zeven <- do.call("estepEEV", c(cv1Modd, list(data=lansing[even,-3])))$z
> compClass(map(cv1Zeven), lansing[even,3])$error
[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 15.

Another option for model selection that is much quicker to compute than crossvalidation is to use `mstep` to fit each model to the training data, then select the best fitting model via BIC. 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:

```
> round(bicEMtrain(lansing[odd,-3], labels = lansing[odd,3]),1)
    EII    VII  EEI    VEI    EVI    VVI    EEE    EEV    VEV    VVV
 -512.1 -526.5 -520 -533.7 -539.5 -554.7 -519.7 -534.3 -549.7 -553.4
```



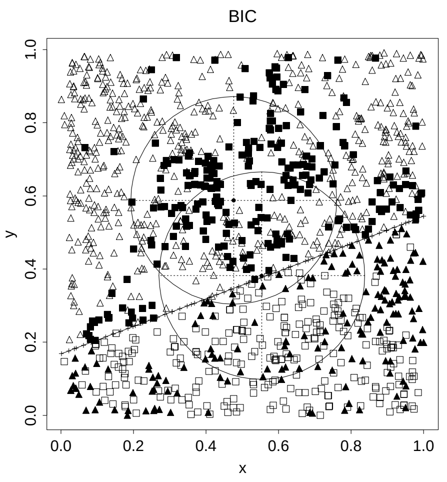


Figure 15: Discriminant analysis using `mstep` and `estep`. The plots show the ellipses corresponding to the standard deviation of each of the two Gaussians used to fit the training data, as well as the discriminant curve (hatched line) where the two groups contribute equally to the mixture density. The discriminant curve would pass through the intersection of the ellipses if they were scaled according to the mixing proportions of model. Filled symbols represent misclassied training [odd-numbered] and test [even-numbered] observations. Left: Model `EEV` 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%:

```
> bicModd <- mstepEII(data=lansing[odd,-3], z=unmap(lansing[odd,3]))
> bicZodd <- do.call("estepEII", c(bicModd, list(data=lansing[odd,-3])))$z
> compClass(map(bicZodd), lansing[odd,3])$error
[1] 0.3513957

> even <- seq(from = 2, to = nrow(lansing), by = 2)
> bicZeven <- do.call("estepEII", c(bicModd, list(data=lansing[even,-3])))$z
> compClass(map(bicZeven), lansing[even,3])$error
[1] 0.3305921
```

The results for this analysis are displayed in the right hand plot of Figure 15.

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

[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.

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.

10.2.1 `mclustDA`

If both training and test sets are given in advance, the function `mclustDA` can be used for discriminant analysis. Its input is the training data and associated class labels, and the test data. Six candidate models are considered in the training phase: `EII`, `VII`, `EEI`, `VVI`, `EEE` and `VVV`. 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.

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

        trainClass mclustModel numGroups
hickory    hickory         EII         4
  maple      maple         EEE         4

 training error rate: 0.228

> compClass(discrim$testClassification, lansing[even,3])$error
[1] 0.2450658
```

The error rates for `mclustDA` classification of the Lansing Woods data are about 23% and 25% for the training [odd-numbered] and test [even-numbered] data, respectively.

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

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

> plot(discrim, trainingData = 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 16 shows the result of selecting `All`. Figure 17 shows the training models for the two classes, as well as the discriminant curve and the misclassified (training and test) data points.

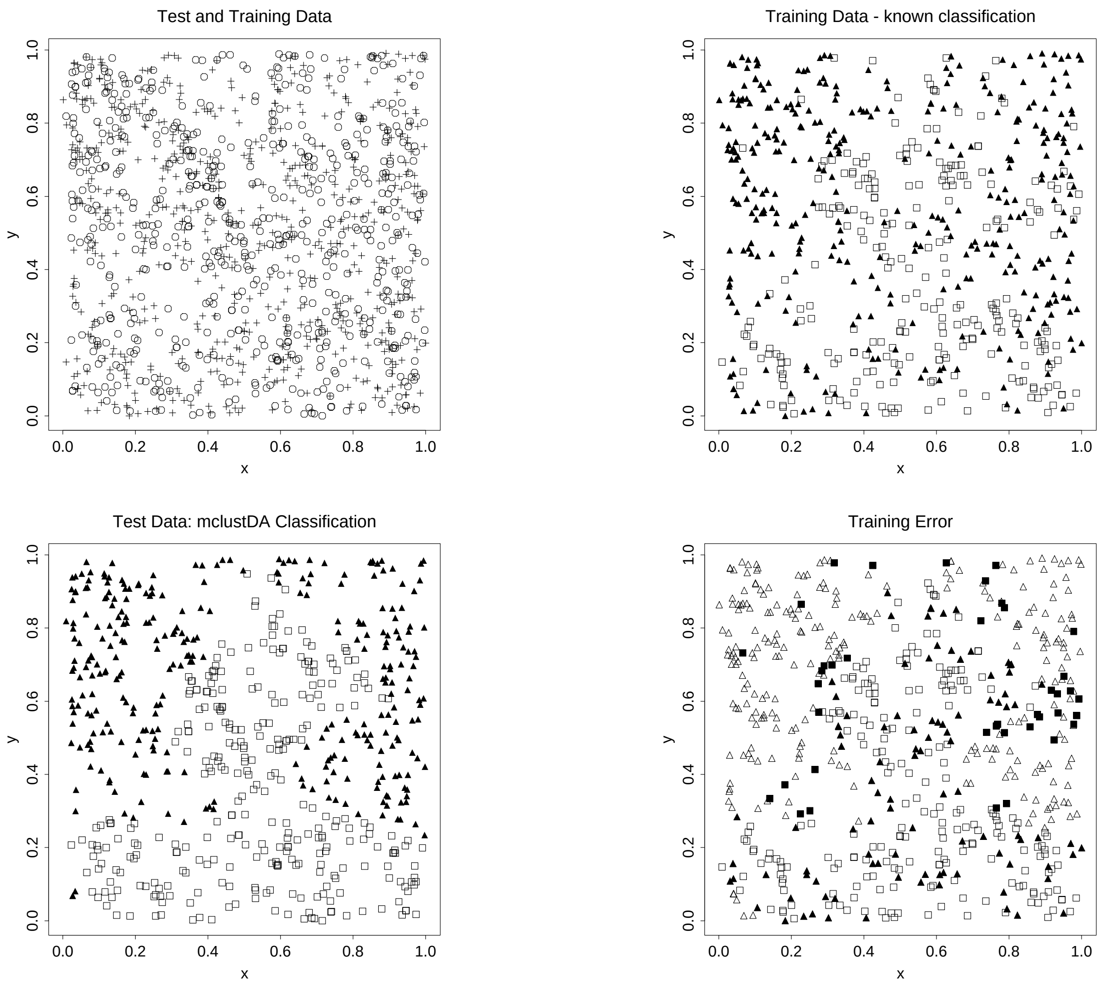


Figure 16: Plots associated with `mclustDA`. Upper Left: the training [odd-numbered/circles] and test [even-numbered/crosses] Lansing Woods data. Upper Right: the training data with known classification. Lower Left: the `mclustDA` classification of the data. Lower Right: the errors (filled symbols) in using the `mclustDA` model to classify the training data.

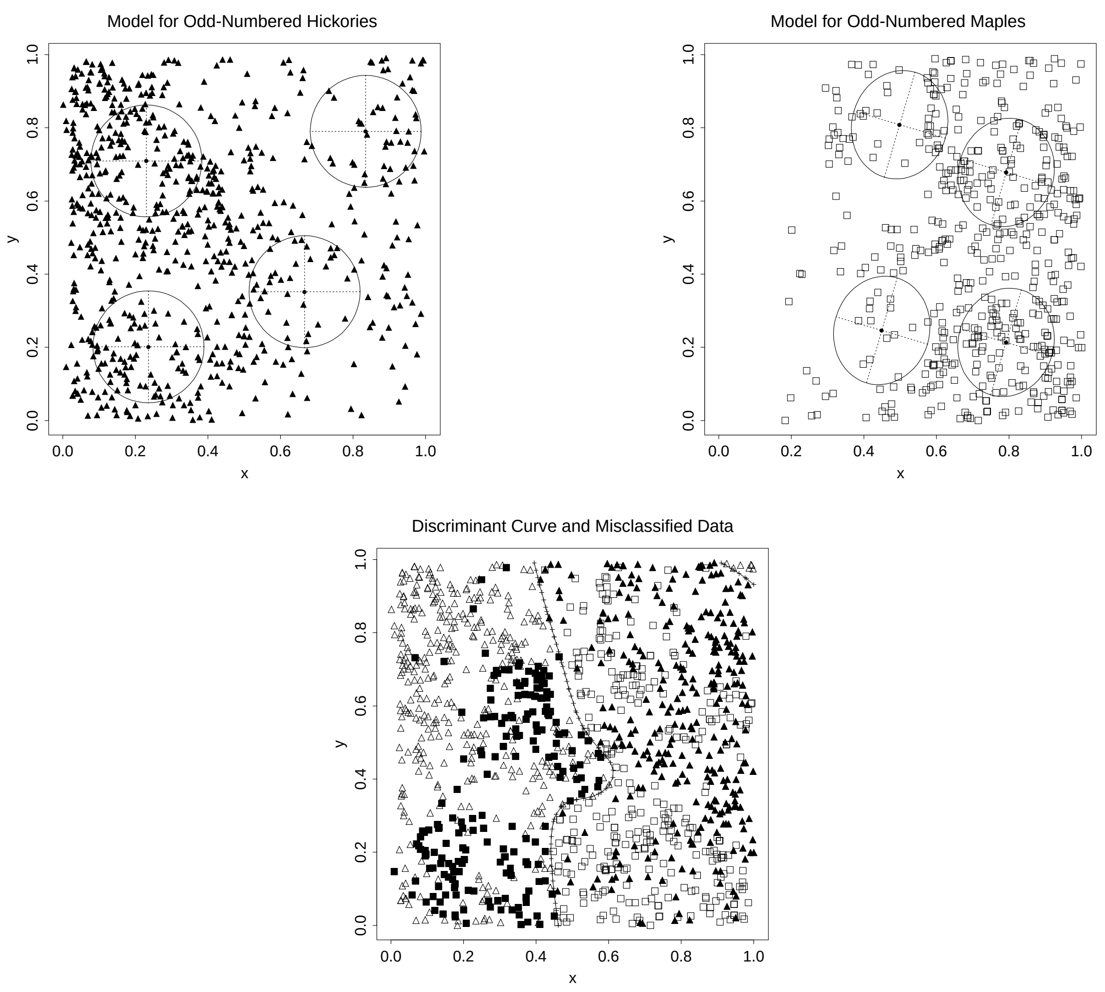


Figure 17: `mclustDA` models and discriminant curve for the Lansing Woods data. Upper Left: the training [odd-numbered] data and its model. Upper Right: the test [even-numbered] data and its model. Lower: discriminant curve with training and test errors (filled symbols). Note the small piece of the curve in the upper right-hand corner.

10.2.2 `mclustDAtrain` 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 that testing, it can be advantageous to separate training and testing computations. In contrast to `mclustDA`, `mclustDAtrain` 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 the training classes are modeled separately instead of modeling the training data as mixture.

```
> train1 <- mclustDAtrain(data=lansing[odd,-3], labels=lansing[odd,3], G=1)
```

```
 EEE EII
   1   1
```

The best model for the hickories has an ellipsoidal covariance, while the best model for the maples has a spherical covariance. The error rates are somewhat higher than those obtained by modeling the training data as a mixture with two components (Section 10.1):

```
> test1odd <- mclustDAtest(data = lansing[odd,-3], models = train1)

> names(summary(test1odd))
[1] "classification" "z"

> compClass(summary(test1odd)$classification, lansing[odd,3])$error
[1] 0.3513957

> test1even <- mclustDAtest(data = lansing[even,-3], models = train1)

> compClass(summary(test1even)$classification, lansing[even,3])$error
[1] 0.3618421
```

Prior probabilities for each class can be supplied to the `summary` function for `mclustDAtest`. For example, the proportions of the known classes in the training set could be used as prior probabilites. In the Lansing Woods example, the error rates are somewhat improved with these prior probabilities:

```
> Hodd <- as.character(lansing[odd, 3]) == "hickory"

> prOdd <- length(odd[Hodd])/length(odd)
> prOdd
[1] 0.5779967

> probs <- c(prOdd, 1-prOdd)

> compClass(summary(test1odd, pro=probs)$classification, lansing[odd,3])$error
[1] 0.3497537

> compClass(summary(test1even,pro=probs)$classification, lansing[even,3])$error
[1] 0.3388158
```

These discriminant analysis results are displayed in Figure 18.

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:

```
> train <- mclustDAtrain(data = lansing[odd,-3], labels = lansing[odd,3])
  EEV EEE
    7   4
```

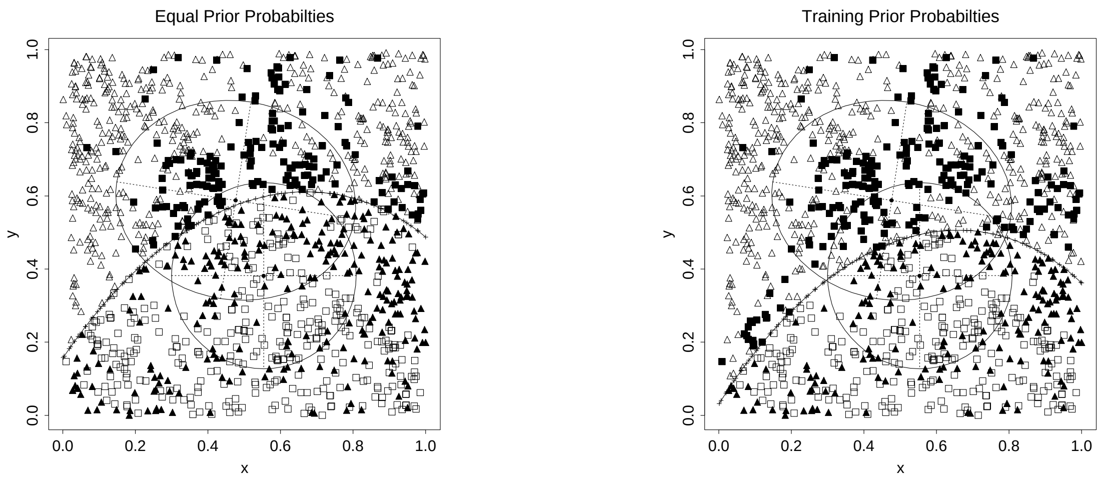


Figure 18: `McIustDA` with one component per training class. The plots show one standard deviation of the Gaussians used to fit each class of the training data, as well as the discriminant curve (hatched line) where the two groups have equal posterior density. Filled symbols represent misclassied training [odd-numbered] and test [even-numbered] observations. Left: Assumes equal prior probabilties. Right: Assumes training prior probabilties.

```
> summary(train)
$hickory:
$hickory$model:
[1] "EEV,7"

$hickory$classification:
  [1] 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 2 2 2 2 1 1 3 3 3 3 3 3 3 3 3 3 3
 [38] 3 3 3 3 2 2 3 2 2 2 2 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 3 2 2 1 1
 [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 2 3 2 2 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 2 2 3 2 3 2 2 3 2 1 1 6 2 2 2 2 4 2 2 4 6 6 1 1 5 6 6 6 4 4 4 2 2 1 1
[149] 1 2 2 2 2 4 2 2 4 6 6 1 1 5 5 6 6 4 4 4 3 3 1 6 1 2 2 2 2 4 2 4 4 6 6 6 1
[186] 5 5 6 6 4 4 4 2 1 1 6 1 2 2 2 2 4 2 4 6 6 6 1 1 5 5 6 6 4 4 4 3 1 1 6 1 2
[223] 2 2 2 4 2 4 6 6 6 1 1 5 5 6 6 4 4 4 4 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7 5 5
[260] 5 5 5 4 5 4 5 5 5 5 7 5 7 7 7 7 7 7 7 5 5 7 5 5 5 5 4 5 5 5 5 5 5 7 7 7 7
[297] 7 7 7 7 7 5 5 5 5 5 5 6 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 5 5 7 5 5 5 5 5 5 5
[334] 5 5 5 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 4
```

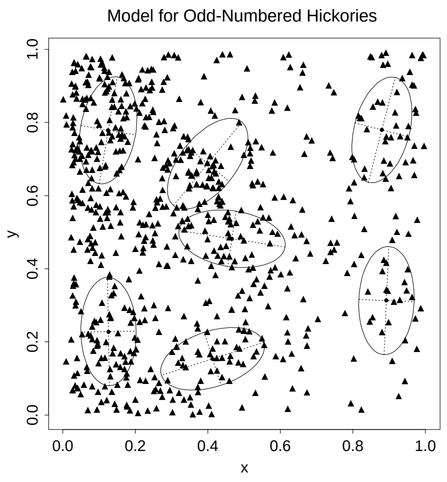
```
$maple:
$maple$model:
[1] "EEE,4"

$maple$classification:
  [1] 1 1 1 2 1 1 1 1 1 1 1 2 2 2 2 2 2 3 2 2 2 4 2 4 1 1 1 2 1 1 1 1 1 1 1 2 2
 [38] 2 2 2 2 3 2 2 2 4 1 4 1 1 1 2 2 1 1 1 1 1 1 2 2 2 2 2 3 3 2 2 2 4 1 4 1 1
 [75] 2 2 1 1 1 1 1 1 1 2 2 2 2 2 3 3 3 2 2 2 1 4 1 1 2 2 1 1 1 1 1 1 1 2 2 2 2
[112] 2 3 3 3 2 2 4 1 4 1 1 4 4 4 4 4 4 2 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 1 4 4
[149] 4 4 4 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 1 4 4 4 4 4 2 3 3 3 3 3 3 3 3 4
[186] 4 4 4 4 4 4 4 1 1 4 4 4 4 4 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 1 4 4 4 4 4
[223] 4 2 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 3 3 4 3 3 3 4 3 3 4 3 3 1 3 3 3
```

In this case `mclustDAtrain` chooses the 7-class `EEV` model for the hickory training class and the 4-class `EII` model (the same model chosen by `mclustDA`) for the maple training class.

The density of the test data 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`:

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

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

> compClass(summary(testTrain)$classification, lansing[odd,3])$error
[1] 0.2249589

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

> compClass(summary(testTest)$classification, lansing[even,3])$error
[1] 0.2450658
```

The error rates are about 23% and 25% for the training [odd-numbered] and test [even-numbered] data, respectively, in this case abput the same as `mclustDA` which restricts the class of models. Figure 19 shows the 7-group `EEV` model for the hickory training class, as well as the discriminant curve and the misclassified (training and test) data points.



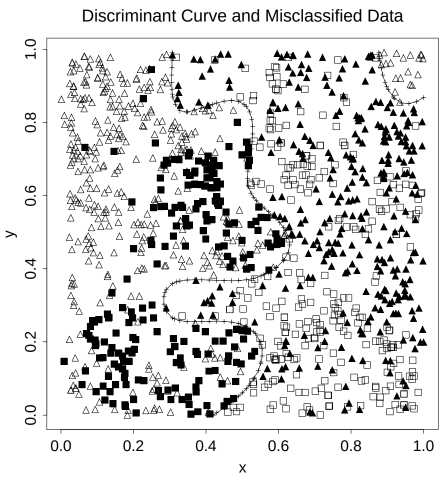


Figure 19: `mclustDAtrain` models and discriminant curve for the Lansing Woods data. Left: the hickory training [odd-numbered] class with its 7-group `EEV` model. Right: discriminant curve with training and test errors (filled symbols).

11 One Dimensional Data

The MCLUST functions for clustering, density estimation and discriminant analysis work on 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:

```
> 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 a special plotting function mclust1Dplot:

```
> par(mfrow = c(2,2))

> xBIC <- EMclust(x)
> plot(xBIC)
   E   V
 "1" "2"

> xModel <- summary(xBIC,x)

> do.call("mclust1Dplot", c(list(data = x), xModel))

mclust1Dplot: make a plot selection (0 to exit):

1: classification
2: uncertainty1: classification
2: uncertainty
3: density
4: all
Selection:
```

Figure 20 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.

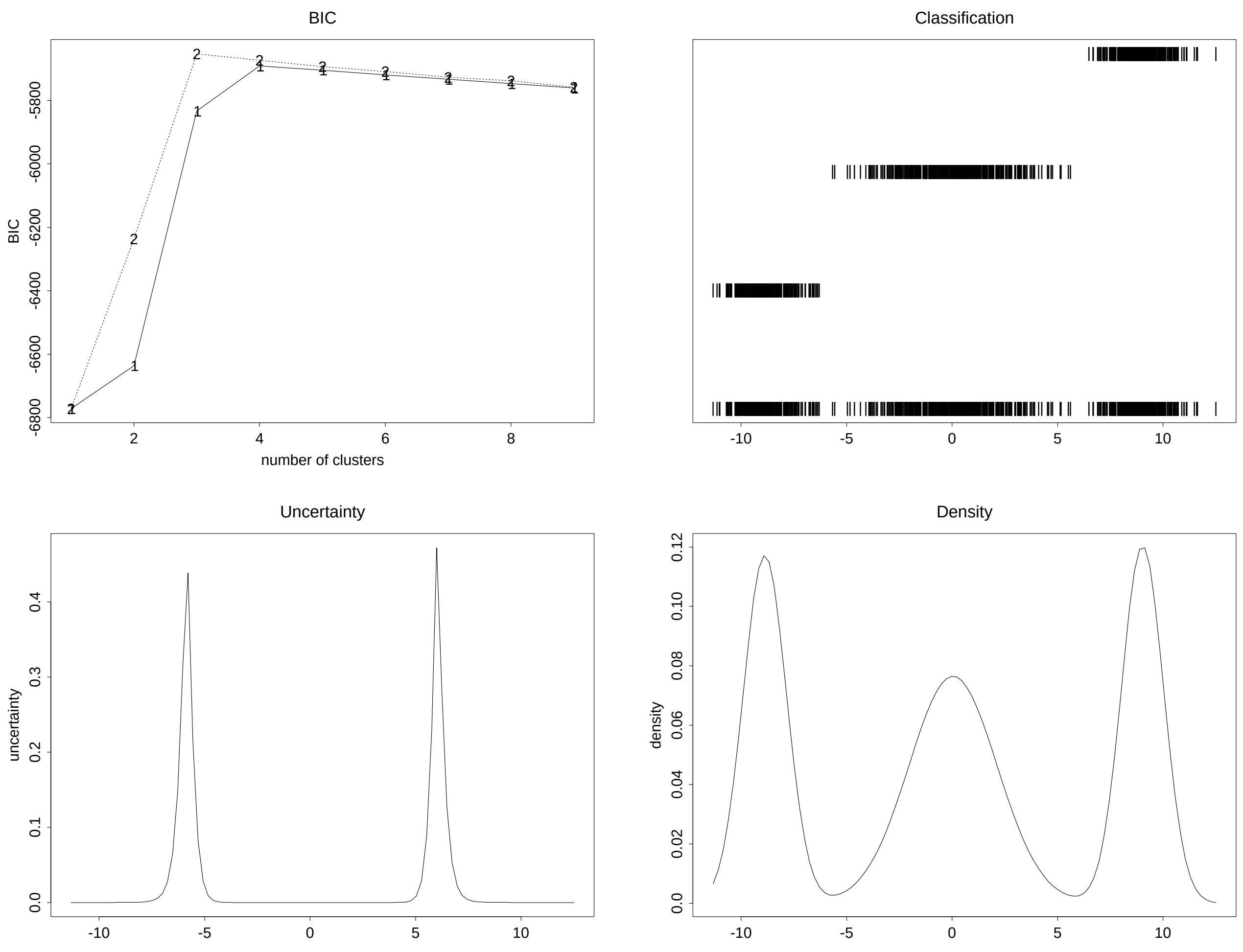


Figure 20: Model-based clustering of one-dimensional data. Clockwise from upper left: BIC, classification, uncertainty, and density from `EMclust` applied to the simulated one-dimensional example. In the classification plot, all of the data is displayed at the bottom, with the separated classes shown different levels above.

```
> 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))
```

We use the following simulated data as a test set:

```
> 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 1
> round(bicEMtrain(x,labels=xClass),3)
         E  V
 -6785.284 NA
```

The training and test errors for the data are as follows:

```
> xMstep <- mstepE(data = x, z = unmap(xClass))

> xZ <- do.call("estepE", c(list(data = x), xMstep))$z
> compClass(map(xZ),xClass)$error  ## training error
[1] 0.4

> yZ <- do.call("estepE", c(list(data = y), xMstep))$z
> compClass(map(yZ),yClass)$error ## testing error
[1] 0.3333333
```

For discriminant analysis via `MclustDA` (Section 10.2):

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

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

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 21. 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.

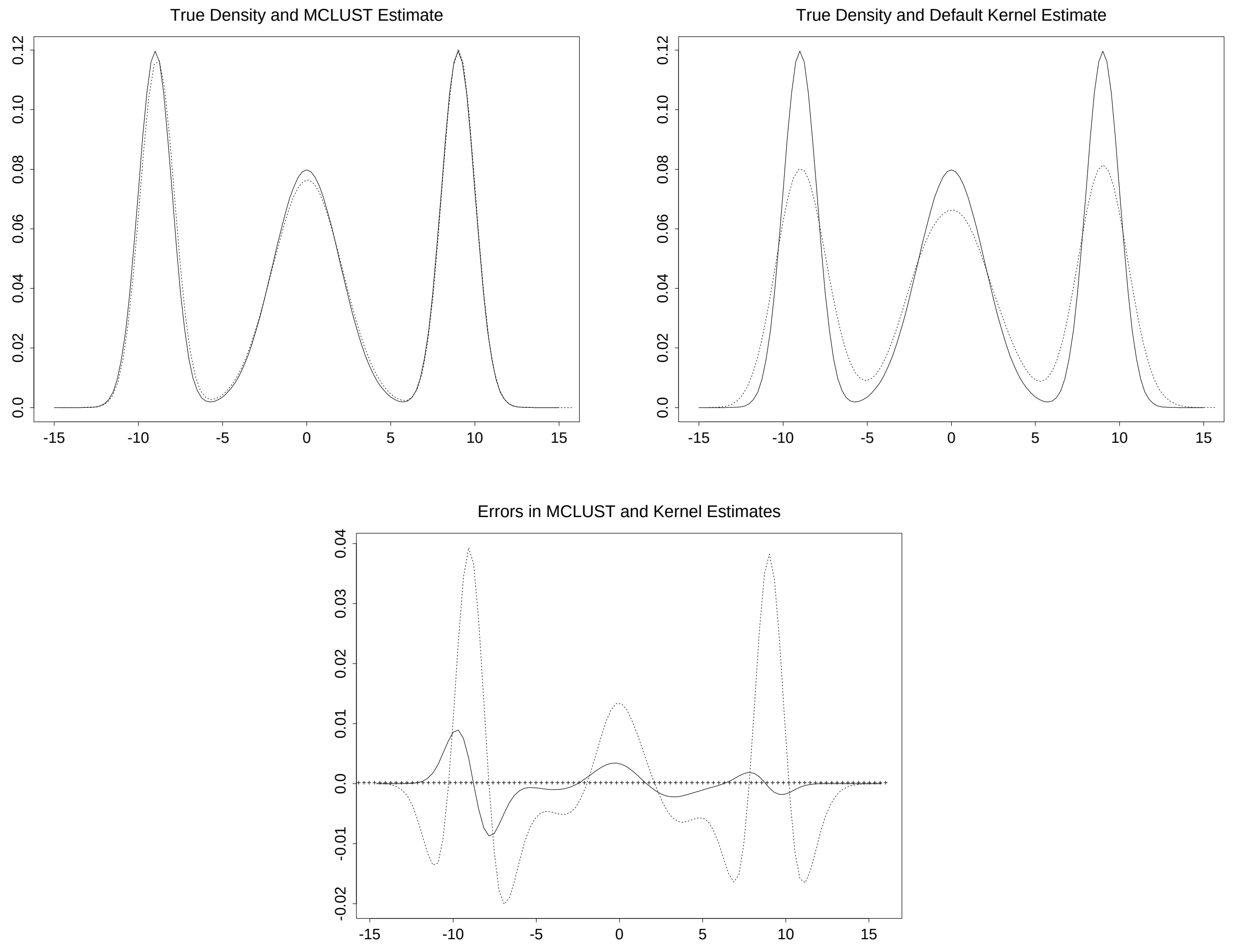


Figure 21: A comparsion of **`method = "mclust"`** and default kernel options for the augmented S-PLUS **`density`** function on the one-dimensional simulated example. Upper Left: The underlying density (solid line) and the computed density with **`method = "mclust"`** (dotted line). Upper Right: The underlying density (solid line) and the computed density with default kernel method (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:

```
> n <- nrow(irisMatrix)
> n
[1] 150
> 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 methodolgy 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 $\mathcal{O}(d^2)$ 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.
`hclass`	Classifications corresponding to `hc` results.

13.2 Parameterized Gaussian Mixture Models

`em`	EM algorithm (starting with E-step).
`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

`.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.
`compClass`	Compare two classifications with equal numbers of classes.
`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

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

13.7.2 Two-Dimensional Data

`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.

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

13.7.4 Other Plotting Functions

`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

[1] D. Allard and C. Fraley. Nonparametric maximum likelihood estimation of features in spatial point processes using Voronoï tessellation. *Journal of the American Statistical Association*, 92:1485–1493, 1997. (available as technical report no. 293R at http://www.stat.washington.edu/www/research/reports).

[2] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[3] A. W. Bowman and A. Azzalini. *Applied Smoothing Techniques for Data Analysis.* Clarendon Press, 1997.

[4] S. D. Byers and A. E. Raftery. Nearest neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93:577–584, 1998.

[5] J. G. Campbell, C. Fraley, F. Murtagh, and A. E. Raftery. Linear flaw detection in woven textiles using model-based clustering. *Pattern Recognition Letters*, 18:1539–1548, 1997. (available as technical report no. 314 at http://www.stat.washington.edu/www/research/reports).

[6] J. G. Campbell, C. Fraley, D. Stanford, F. Murtagh, and A. E. Raftery. Model-based methods for real-time textile fault detection. *International Journal of Imaging Systems and Technology*, 10:339–346, 1999.

[7] G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28:781–793, 1995.

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

[9] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[10] C. Fraley and A. E. Raftery. MCLUST: Software for model-based cluster analysis. *Journal of Classification*, 16:297–306, 1999.

[11] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2002. (available as technical report no. 329 at http://www.stat.washington.edu/www/research/reports).

[12] H. P. Friedman and J. Rubin. On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62:1159–1178, 1967.

[13] D. J. Gerrard. Competition quotient: a new measure of the competition affecting individual forest trees. Research Bulletin No. 20, Agricultural Experimental Station, Michigan State University, 1969.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins, 3rd edition, 1996.

[15] S. P. Kaluzny, S. C. Vega, T. P. Cardoso, and A. A. Shelly. *S+SpatialStats: User's manual for Windows and UNIX.* Springer, 1998.

[16] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis.* Academic Press, 1979.

[17] F. Murtagh and A. E. Raftery. Fitting straight lines to point patterns. *Pattern Recognition*, 17:479–483, 1984.

[18] K. Roeder and L. Wasserman. Practical Bayesian density estimation using mixtures of normals. *Journal of the American Statistical Association*, 92:894–902, 1997.

[19] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[20] A. J. Scott and M. J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, 27:387–397, 1971.

[21] J. H. Ward. Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, 58:234–244, 1963.