MCLUST Version 3 for R: Normal Mixture Modeling and Model-Based Clustering*†

Chris Fraley and Adrian E. Raftery

Technical Report No. 504

Department of Statistics
University of Washington
Box 354322
Seattle, WA 98195-4322 USA

September 2006
revised: 1/2007, 11/2007, 11/2009, 12/2009, 7/2010, 01/2012

MCLUST is a contributed R package for normal mixture modeling and model-based clustering. It provides functions for parameter estimation via the EM algorithm for normal mixture models with a variety of covariance structures, and functions for simulation from these models. Also included are functions that combine model-based hierarchical clustering, EM for mixture estimation and the Bayesian Information Criterion (BIC) in comprehensive strategies for clustering, density estimation and discriminant analysis. There is additional functionality for displaying and visualizing the models along with clustering and classification results. A number of features of the software have been changed in this version, and the functionality has been expanded to include regularization for normal mixture models via a Bayesian prior. MCLUST is licensed by the University of Washington and distributed through CRAN; see http://cran.r-project.org/web/packages/mclust/index.html.

*Funded by National Institutes of Health grant 8 R01 EB002137–02 and by Office of Naval Research grant N00014-01-1-0745.

†Many users have offered comments and suggestions leading to improvements in the MCLUST documentation and software over the years. Christian Hennig and Ron Wehrens deserve special thanks in this regard. We are also indebted to Ron Wehrens for porting and maintaining earlier versions of MCLUST for R.

Contents

List of Tables

List of Figures

1 Overview

The MCLUST software [10, 12] currently includes the following features:

- Model-based clustering (model and number of clusters selected via BIC).
- Normal mixture modeling via EM for ten covariance structures.
- Simulation from parameterized Gaussian mixtures.
- Discriminant analysis via MclustDA.
- Model-based hierarchical clustering for four covariance structures.
- Displays, including uncertainty plots and random projections.

This manuscript describes Version 3 of MCLUST for R, which allows regularization in normal mixture models via a Bayesian prior [15]. A number of other aspects of the software have been changed as well, to reflect evolution in its use. A comprehensive treatment of the methods used in MCLUST can be found in [11, 15].

MCLUST is available as a contributed package (mclust) in the R language. It can be obtained from CRAN at http://cran.r-project.org/web/packages/mclust/index.html. Follow the instructions for installing R packages on your machine, and then do

```
> library(mclust)
```

inside R in order to use the software. Throughout this manual it will be assumed that these steps have been taken before running the examples.

MCLUST is licensed by the University of Washington, and users are expected to comply with the license agreement accompanying the software. Users of packages that invoke MCLUST are also expected to comply with the terms MCLUST license.

Several contributed R packages have functions that require MCLUST, including clustvarcel, fpc, prabclus, and the Bioconductor package spotSegmentation [13]. A couple of tutorials on mclust have also been published [14, 16].

2 Model-Based Cluster Analysis

MCLUST provides functionality for cluster analysis combining model-based hierarchical clustering (section 5), EM for Gaussian mixture models (section 3), and BIC (section 4).

2.1 Basic Cluster Analysis Example using Mclust

As an illustration, consider the bivariate faithful dataset (included in the R language distribution) shown in Figure 1. The following command performs a cluster analysis of the faithful dataset, and prints a summary of the result:

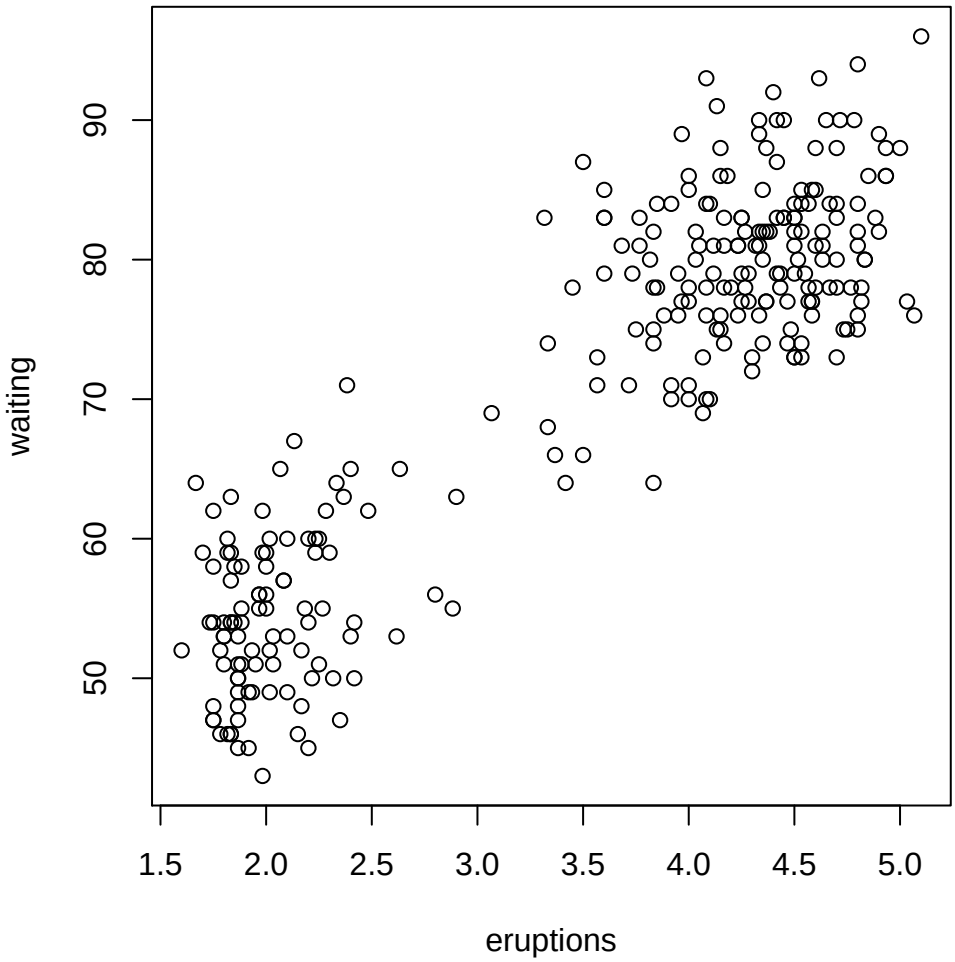

Figure 1: The bivariate `faithful` dataset.

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

identifier	Model	HC	EM	Distribution	Volume	Shape	Orientation
E		•	•	(univariate)	equal		
V		•	•	(univariate)	variable		
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	$\lambda D A D^T$	•	•	Ellipsoidal	equal	equal	equal
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
VVV	$\lambda_k D_k A_k D_k^T$	•	•	Ellipsoidal	variable	variable	variable

```
> faithfulMclust <- Mclust(faithful)
> faithfulMclust

 best model: EEE with 3 components
```

In this case, the best model according to BIC is an equal-covariance model with 3 components or clusters. The clustering results can be displayed as follows:

```
> plot(faithfulMclust, data = faithful)
```

The corresponding plots are shown in Figure 2. The covariance structures defining the models available in MCLUST are summarized in Table 1; these models are explained in more detail in appendix A.

The input to function Mclust includes the number of mixture components and the covariance structures to consider. By default, Mclust compares BIC values for parameters optimized for up to nine components and all ten covariance structures currently available in the MCLUST software. The output includes the parameters of the maximum-BIC model (where the maximum is taken over all of the models and numbers of components considered), and the corresponding classification and uncertainty.

The object produced by Mclust is a list with a number of elements describing the selected model. The names of these elements can be displayed as follows:

```
> names(faithfulMclust)
 [1] "modelName"      "n"              "d"              "G"
 [5] "BIC"            "bic"            "loglik"         "parameters"
 [9] "z"              "classification" "uncertainty"
```

A detailed description is provided in the Mclust help file.

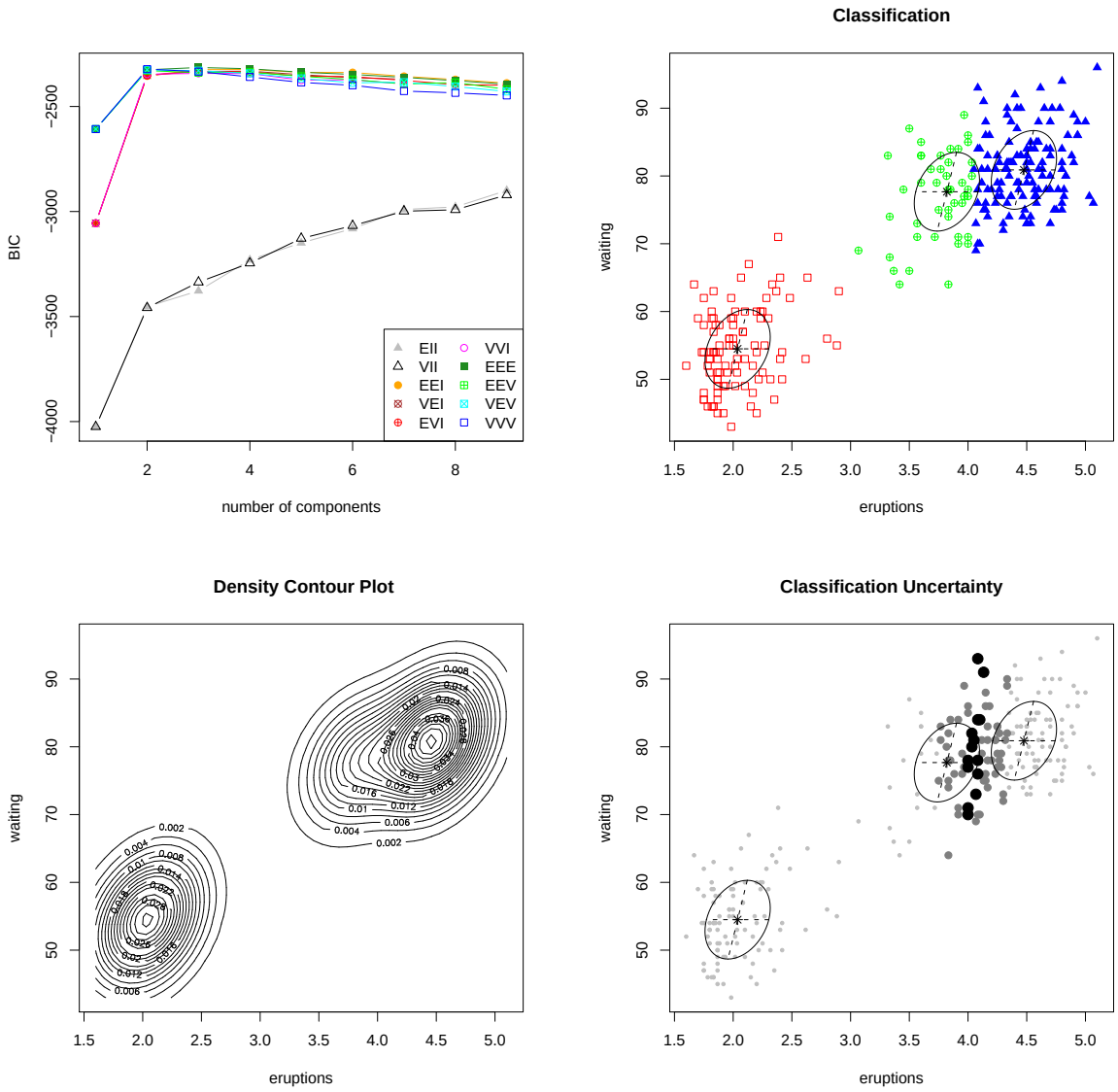

Figure 2: Plots associated with the function `Mclust` for the `faithful` dataset with the default arguments. Clockwise from upper left: BIC, classification, uncertainty, density. The ellipses superimposed on the classification and uncertainty plots correspond to the covariances of the components.

2.2 mclustBIC and its summary function

To do further analysis on the same dataset, for example to see the results for a different set of models and/or different numbers of components, `Mclust` could be rerun. However this approach could involve unnecessary repetition of computations and could also take considerable time when the dataset is large or the process is to be repeated many times. An alternative approach is to split the analysis into several parts using function `mclustBIC`.

For the `faithful` dataset, the following sequence of commands produces the same clustering result as the call to `Mclust`.

```
> faithfulBIC <- mclustBIC(faithful)
> faithfulSummary <- summary(faithfulBIC, data = faithful)
> faithfulSummary
classification table:
  1   2   3
130  97  45

best BIC values:
    EEE,3     EEE,4     VVV,2
-2314.386 -2320.207 -2322.192
```

Although the method used for printing is different, `faithfulSummary` has the same component names as `faithfulMclust`, except that it does not include `"BIC"`, the table of BIC values, which comprise the object `faithfulBIC` computed by `mclustBIC`:

```
> faithfulBIC

 BIC:
        EII       VII       EEI       VEI       EVI       VVI       EEE
1 -4024.721 -4024.721 -3055.835 -3055.835 -3055.835 -3055.835 -2607.623
2 -3452.998 -3458.300 -2354.601 -2350.607 -2352.618 -2346.065 -2325.220
3 -3377.712 -3336.542 -2323.008 -2332.698 -2332.204 -2342.371 -2314.386
4 -3230.246 -3245.732 -2323.676 -2331.829 -2334.756 -2343.068 -2320.207
5 -3149.389 -3128.214 -2337.730 -2348.284 -2355.885 -2374.251 -2336.967
6 -3081.401 -3067.580 -2338.116 -2363.073 -2357.745 -2372.728 -2347.296
7 -2990.334 -2998.496 -2356.458 -2370.071 -2375.850 -2393.086 -2361.216
8 -2978.088 -2991.847 -2371.814        NA -2395.992        NA -2376.920
9 -2899.778 -2920.951 -2388.617        NA -2399.085        NA -2393.733
        EEV       VEV       VVV
1 -2607.623 -2607.623 -2607.623
2 -2329.116 -2325.416 -2322.192
3 -2338.986 -2329.352 -2333.894
4 -2336.750 -2342.472 -2359.216
5 -2366.985 -2367.785 -2390.985
6 -2371.741 -2387.155 -2398.905
7 -2392.961 -2391.166 -2426.431
```

```
8 -2404.598 -2404.932 -2437.612
9 -2427.039 -2428.375 -2449.787
```

The missing values are models and numbers of clusters for which parameter values could not be fit (using the default initialization). For multivariate data, the default initialization for all models uses the classification from hierarchical clustering based on an unconstrained model. For univariate data, the default is to divide the data into quantiles for initialization.

The `summary` method for `mclustBIC` allows specification of the models and numbers of clusters over which the best model is to be chosen, allowing models other than the maximum BIC model to be extracted and analyzed.

The `plot` method for `mclustBIC` allows specification of the models and numbers of clusters, arguments to the `legend` function, as well as setting limits on the vertical axis. For example, the following shows the maximum BIC values in more detail than the default:

```
> plot(faithfulBIC, G = 1:7, ylim = c(-2500,-2300)
  legendArgs = list(x = "bottomright", ncol = 5))
```

The resulting plot is shown in Figure 3.

2.3 Extended Cluster Analysis Example

As an example of an extended analysis, consider the `wreath` data shown in Figure 4. There are 1000 bivariate observations simulated from a 14-component model in which the component covariance matrices are of equal size and shape, but differ in orientation. The BIC values can be obtained with a call to `mclustBIC` and then plotted:

```
> data(wreath)
> wreathBIC <- mclustBIC(wreath)
> plot(wreathBIC, legendArgs = list(x = "topleft"))
```

Refering to the BIC plot (shown on the left in Figure 5), the maximum BIC appears to be outside the range of the default values for the number of components in `mclustBIC` (and `Mclust`). More components (for example, up to 20) can be considered in the analysis without recomputing previous results:

```
> wreathBIC <- mclustBIC(wreath, G = 1:20, x = wreathBIC)
> plot(wreathBIC, G = 10:20, legendArgs = list(x = "bottomleft"))
> summary(wreathBIC, wreath)
```

The BIC plot is shown on the right in Figure 5. Using `summary` to obtain the best model according to BIC, a 14-component `EEV` model is chosen, which is in agreement with how the data was simulated.

```
> wreathModel <- summary(wreathBIC, data = wreath)
> wreathModel

classification table:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14
```

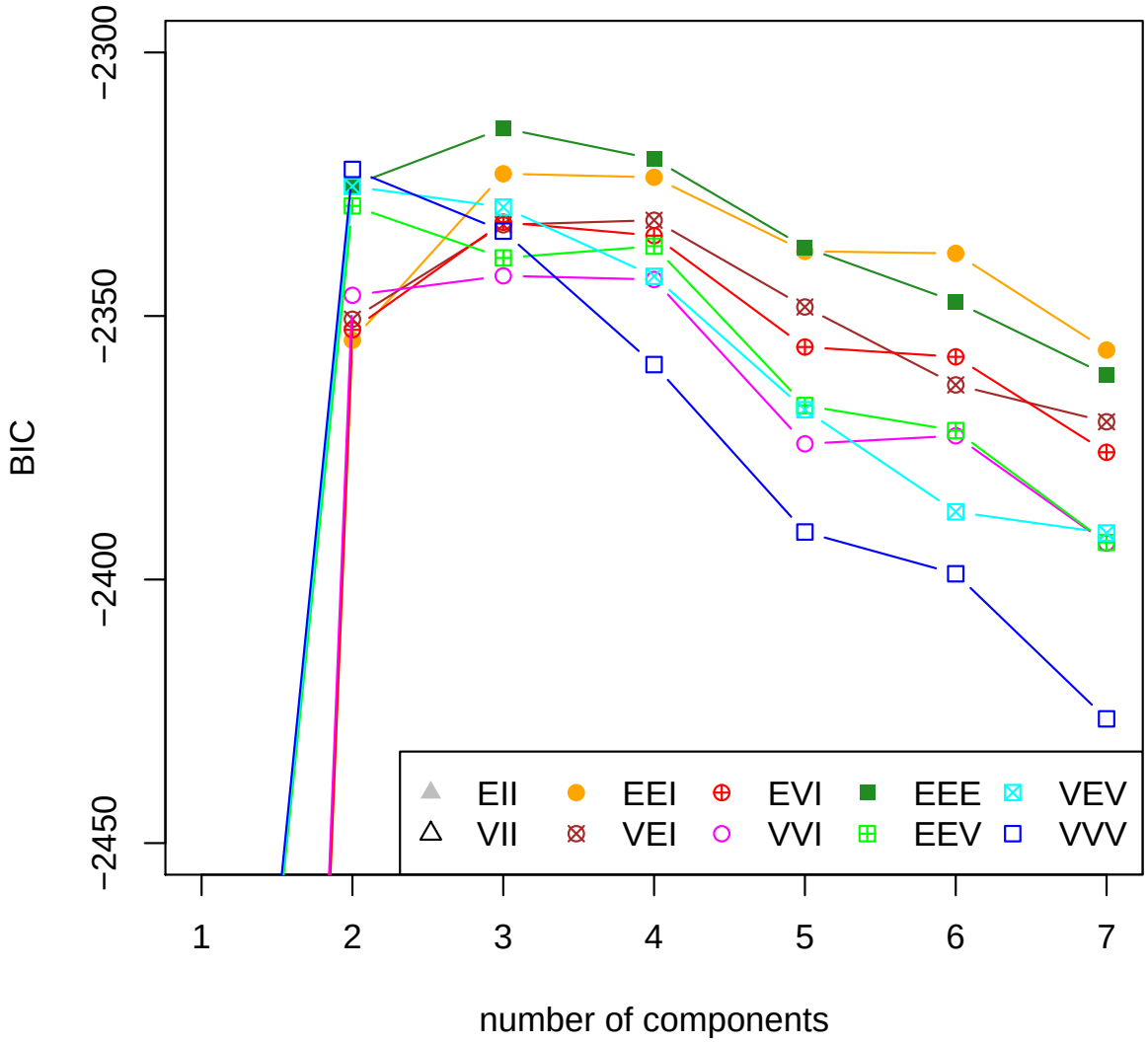

Figure 3: BIC plot for the `faithful` dataset, with vertical axes adjusted to display the maximum values.

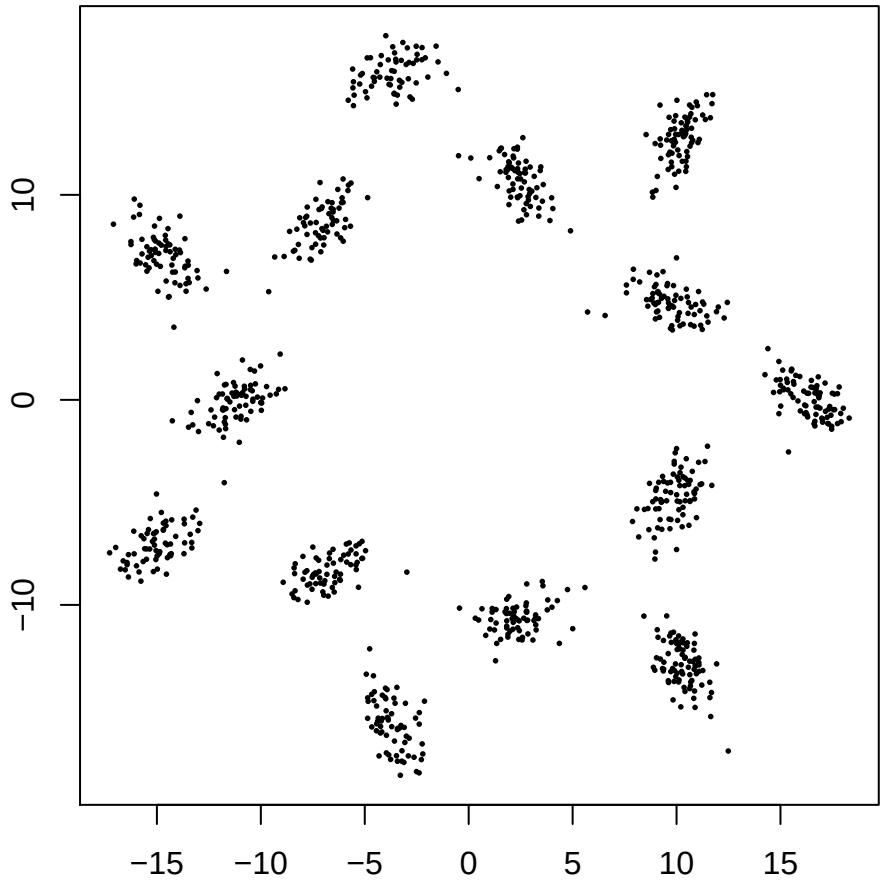

Figure 4: The bivariate `wreath` dataset, which consists of 1000 observations simulated from a 14-component normal mixture in which the component covariance matrices are of equal size and shape, but differ in orientation.

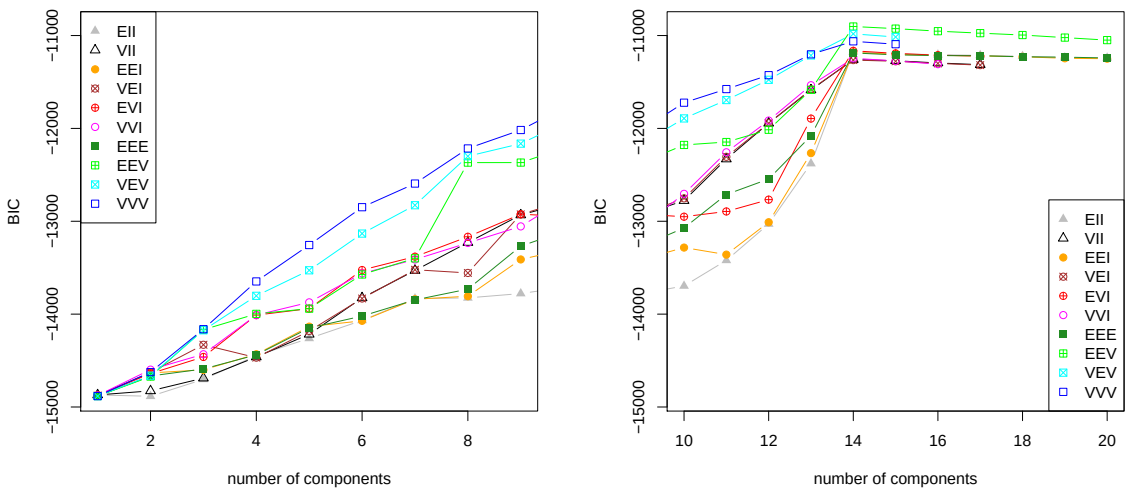

Figure 5: BIC for `wreath` dataset. LEFT: BIC for all models and up to 9 components (the default in `mclustBIC` and `Mclust`). RIGHT: BIC for 10:20 components, all models. There is a clear peak for all models at 14 components.

```
74 69 63 74 68 70 71 66 83 77 66 77 61 81

best BIC values:
   EEV,14    EEV,15    EEV,16
-10902.77 -10919.96 -10944.09
```

The model for the `wreath` dataset is shown in Figure 6. The `summary` function can also be used to restrict the set of models and/or numbers of clusters over which the best model is chosen according to BIC. For example, the following commands produce the best spherical model for the `wreath` data:

```
> wreathSphericalModel <- summary(wreathBIC, data = wreath,
                                  modelNames = c("EII", "VII"))
> wreathSphericalModel

classification table:
 1  2  3  4  5  6  7  8  9 10 11 12 13 14
75 69 63 74 68 70 71 65 83 77 66 77 61 81

best BIC values:
   EII,14    EII,15    EII,16
-11175.90 -11186.51 -11200.04
```

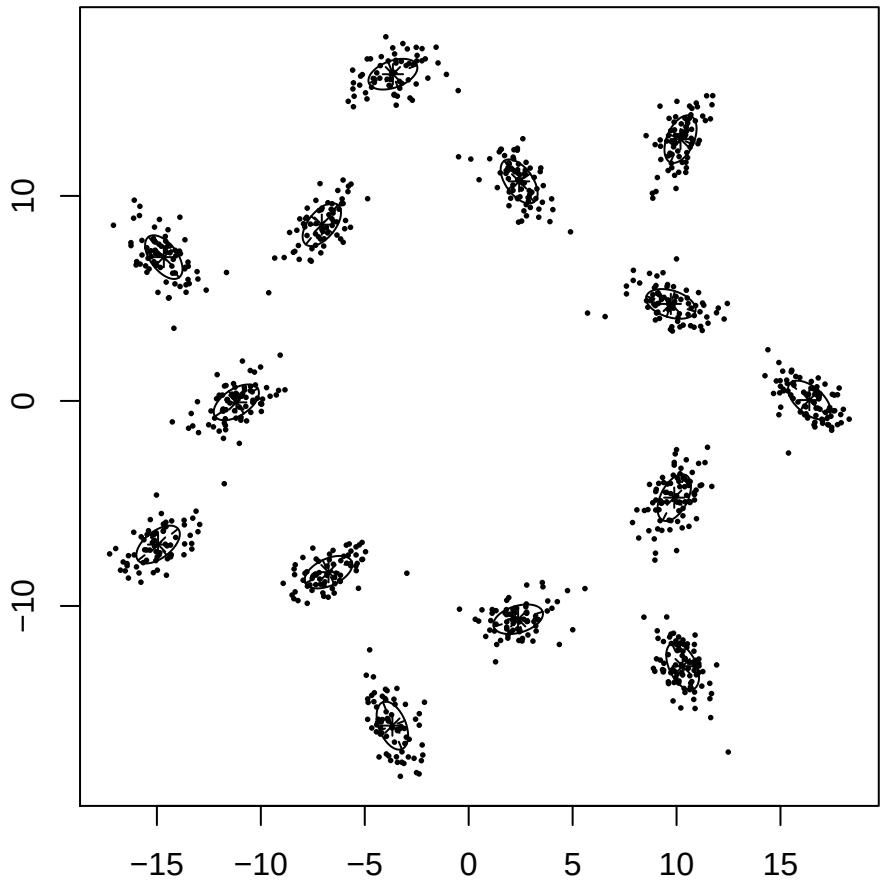

Figure 6: The 14-component `EEV` (equal size and shape) model obtained for the `wreath` dataset. The ellipses superimposed on the plot correspond to the covariances of the components.

2.4 Regularizing with a Prior

It is now possible in MCLUST to specify a prior distribution to regularize the fit to the data [15]. We illustrate the use of a prior on the trees dataset (included in the R language distribution), for which a pairs plot is shown in Figure 7.

The following commands compute and plot the BIC curves for the trees dataset provided in R with and without a prior. Without the prior, the BIC plot shows a number of jagged peaks, and many BIC values are missing for some models due to failure in the EM computations caused by singularity and/or shrinking components. With the prior, the BICs are smoother and there are fewer failures in estimation. See Figure 8.

```
> treesBIC <- mclustBIC(trees) # default (no prior)
> plot(treesBIC, legendArgs = list(x = "bottom", ncol = 2, cex = .75))
> treesBICprior <- mclustBIC(trees, prior = priorControl())
> plot(treesBICprior, legendArgs = list(x = "bottom", ncol = 2, cex = .75))
```

A function priorControl is provided in MCLUST for specifying the prior and its parameters. When called with its defaults, it invokes another function called defaultPrior which can serve as a template for specifying alternative priors. An example of the result of a call to defaultPrior is shown below.

```
> defaultPrior(trees, G=2, modelName = "VVV")
$shrinkage
[1] 0.01

$mean
   Girth   Height   Volume
13.24839 76.00000 30.17097

$dof
[1] 5

$scale
           Girth   Height    Volume
Girth   6.203797  6.54109  31.42755
Height  6.541090 25.57640  39.47333
Volume 31.427545 39.47333 170.21710
```

For more detail on the prior and its specification, see Section A.3.

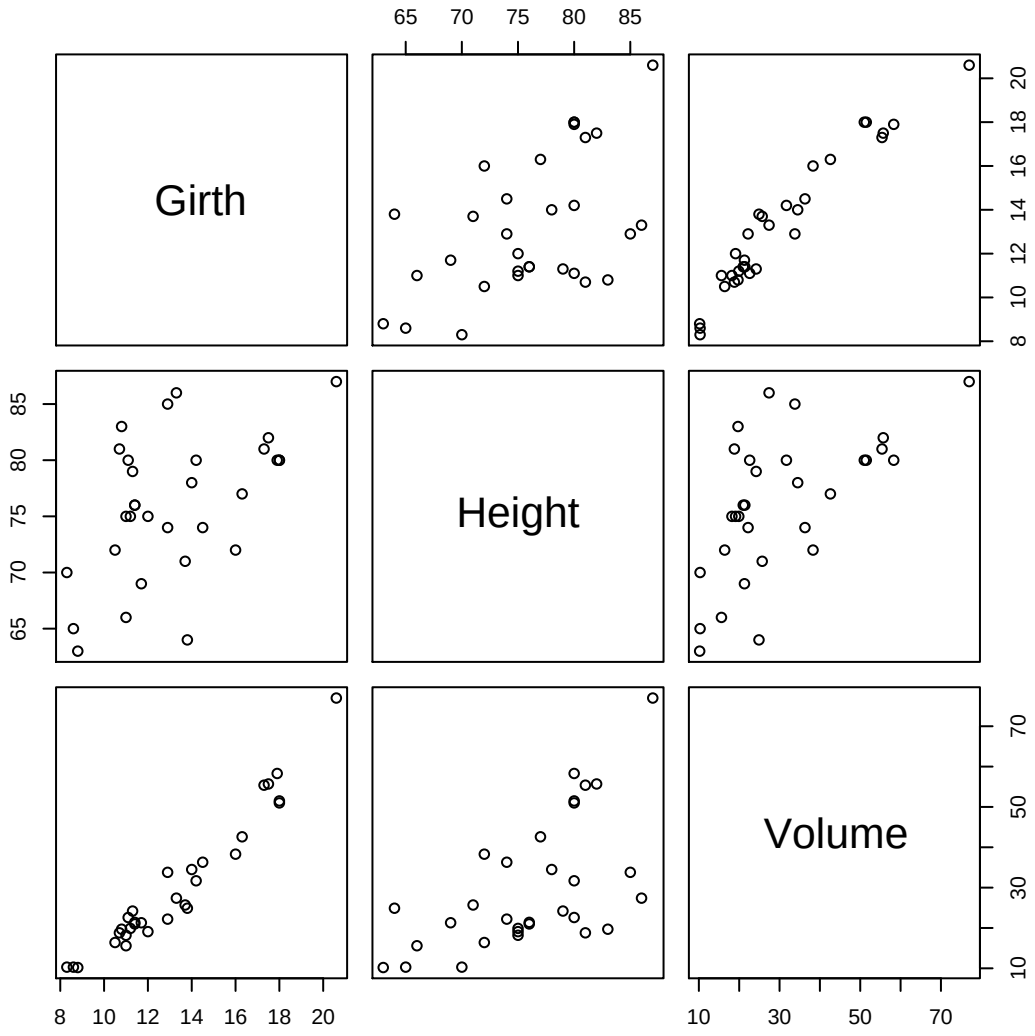

Figure 7: Pairs plot of `trees` dataset.

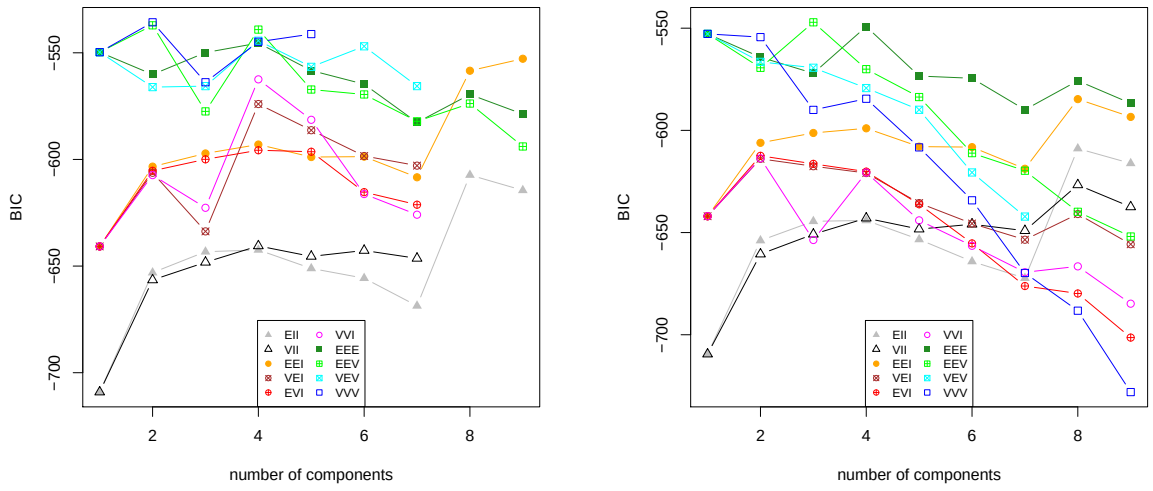

Figure 8: BIC without (left) and with the prior for the `trees` dataset.

2.5 Clustering with Noise and Outliers

MCLUST allows model-based clustering with noise, namely outlying observations that do not belong to any cluster. To include noise in the modeling, an initial guess of the noise observations must be supplied via the `noise` component of the `initialization` argument in `Mclust` or `mclustBIC`. The model for noise used in MCLUST is discussed in more detail in Section A.1 of the appendix, along with some some strategies for obtaining an initial noise estimate.

In the following example, Poisson noise is added to the `faithful` dataset. A random initial estimate was used for noise for the purposes of illustration. This happens to work well in this instance, although we don't recommend this as a general strategy.

```
> b <- apply( faithful, 2, range)
> nNoise <- 500
> set.seed(0)
> poissonNoise <- apply(b, 2, function(x, n)
                        runif(n, min = min(x)-.1, max = max(x)+.1), n = nNoise)
> faithfulNdata <- rbind(faithful, poissonNoise)
> set.seed(0)
> faithfulNoiseInit <- sample(c(TRUE,FALSE),size=nrow(faithful)+nNoise,
                                  replace=TRUE,prob=c(3,1))
> faithfulNbic <- mclustBIC(faithfulNdata,
                            initialization = list(noise = faithfulNoiseInit))
> faithfulNsummary <- summary(faithfulNbic, faithfulNdata)
> faithfulNsummary

classification table:
  0   1   2
521 143 108

best BIC values:
    EVI,2     VVI,2     EEI,2
-7996.437 -7998.035 -8000.251
```

The results are shown in Figure 9. The classification and BIC plots were obtained with the following commands.

```
mclust2Dplot(faithfulNdata, classification=faithfulNsummary$classification,
             parameters=faithfulNsummary$parameters)
plot(faithfulNbic, legendArgs = list(x = "bottomleft", horiz = FALSE,
                                     ncol = 5, cex = 0.75))
```

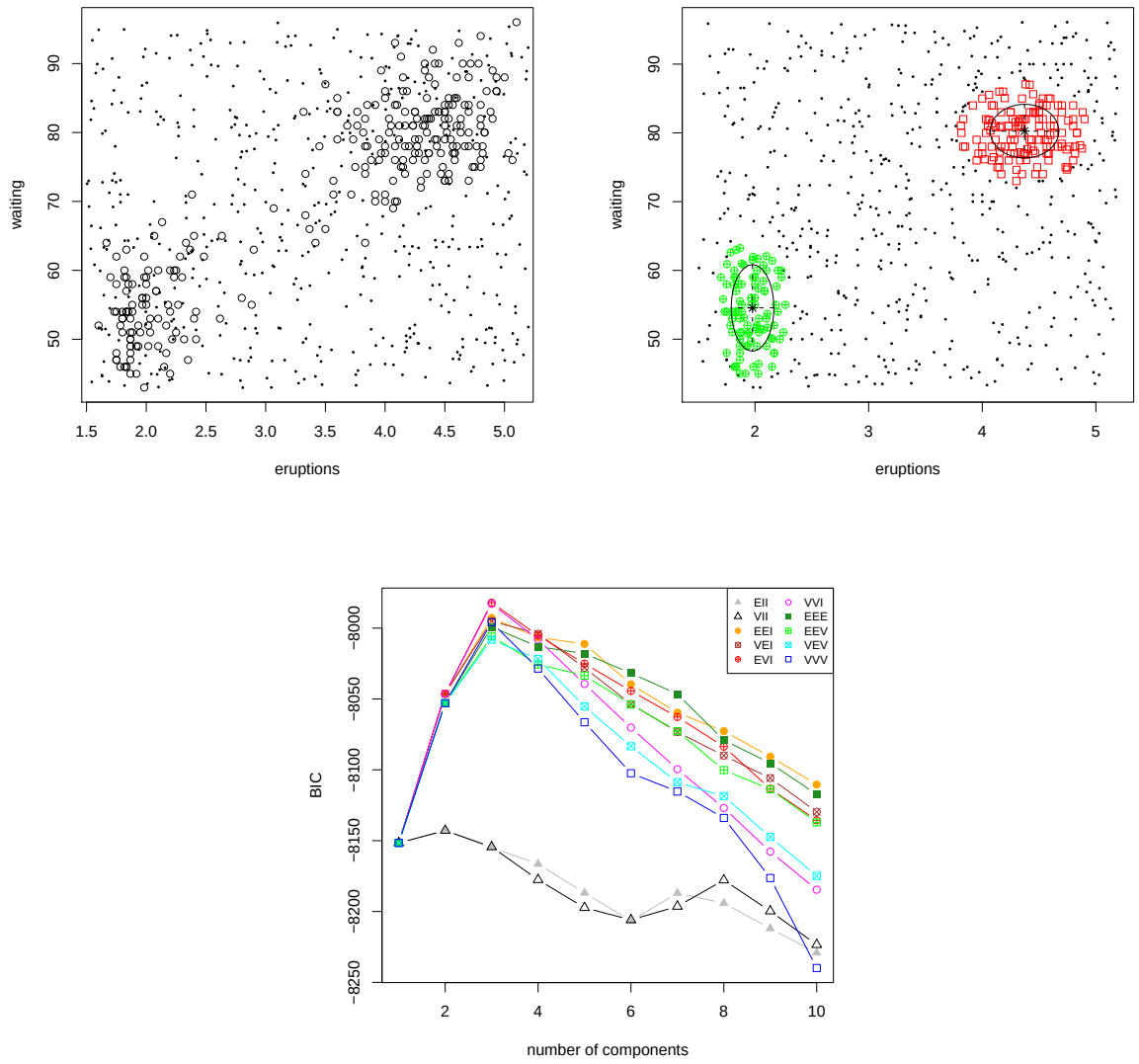

Figure 9: Cluster analysis of the `faithful` dataset with added Poisson noise. Upper Left: The 272 obervations of the `faithful` dataset (circles) with 500 Poisson noise points (small dots). Upper Right: `MCLUST` classification starting with random noise estimate. Lower: BIC.

2.6 Further Considerations in Cluster Analysis

Clustering can be affected by parameter settings such as convergence tolerances within the clustering functions, although the defaults are often adequate. It is also possible do model-based clustering starting with parameter estimates, conditional probabilities, or classifications other than those produced by model-based hierarchical clustering. The functions provided for mixture estimation (Section 3) and BIC (Section 4) can be used for this purpose.

Finally, it is important to take into account numerical issues in cluster analysis. The computations for estimating the model parameters break down when the covariance corresponding to one or more components becomes ill-conditioned (singular or nearly singular). Including a prior (Section A.3) is often helpful in such situations. In general the modeling computations cannot proceed if clusters contain only a few observations or if the observations they contain are very nearly colinear. Computations may also fail when one or more mixing proportions shrink to negligible values. The EM functions in `MCLUST` compute and monitor the conditioning of the covariances, and an error condition is issued (unless such warnings are turned off) when the associated covariance appears to be nearly singular, as determined by a threshold with the default value `emControl()$eps`.

3 EM for Mixture Models

`MCLUST` provides iterative EM (Expectation-Maximization) methods for maximum-likelihood estimation in parameterized Gaussian mixture models. In the models considered here, 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 parameter estimates given z.

`MCLUST` functions `em` and `me` implement the EM algorithm for parameterized Gaussian mixtures. Function `em` starts with the E-step; besides the data and model specification, the model parameters (means, covariances, and mixing proportions) proportions must be provided. Function `me` starts with the M-step; besides the data and model specification, the conditional probabilities z must be provided. The output for both are the maximum-likelihood estimates of the model parameters and z.

3.1 Individual `E` and `M` Steps

Functions `estep` and `mstep` implement 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`. Below we apply `mstep` and `estep` to the `iris` dataset (included in the `R` language distribution).

```
> ms <- mstep( modelName = "VVV", data = iris[,-5], z = unmap(iris[,5]))
> names(ms)
[1] "modelName"  "prior"      "n"          "d"          "G"
[6] "z"          "parameters"

> es <- estep( modelName = "VVV", data = iris[,-5],
```

```
               parameters = ms$parameters)
> names(es)
[1] "modelName"  "n"          "d"          "G"          "z"
[6] "parameters" "loglik"
```

In this example, the initial estimate of z for the M-step is a matrix of indicator variables corresponding to a discrete classification (`iris[,5]`). The function `unmap` converts a discrete classification into the corresponding indicator variables. `MCLUST` allows specification of a prior, for which the EM algorithm will compute a posterior mode. See Sections 2.4 and A.3 for more details. In Section 9.1, we show how to use `mstep` and `estep` for discriminant analysis.

3.2 Uncertainty

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

```
> meVVViris <- me(modelName = "VVV", data = iris[,-5], z = unmap(iris[,5]))
> uncer <- 1 - apply( meVVViris$z, 1, max)
```

The `R` function `quantile` applied to the uncertainty gives a measure of the quality of the classification.

```
> quantile(uncer)
          0%          25%          50%          75%         100%
0.000000e+00 0.000000e+00 1.907041e-08 1.392060e-03 3.361880e-01
```

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 function `uncerPlot`, as is done below for the `iris` example (see Figure 10):

```
> uncerPlot(z = meVVViris$z, truth = iris[,5])
```

It is also possible to plot an uncertainty curve for one-dimensional data (see Section 10) or an uncertainty surface for bivariate data (see Section 6.1).

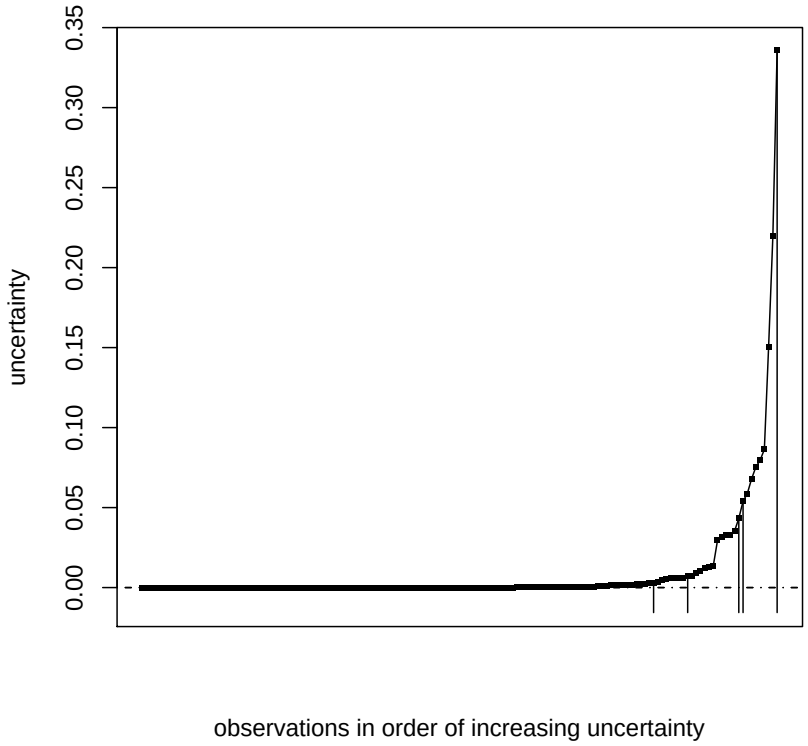

Figure 10: Uncertainty plot for the the 3-cluster mixture model fit of the `iris` dataset via EM based on unconstrained Gaussian mixtures. The vertical lines indicate misclassified observations. The plot was created with function `uncerPlot`, and shows the relative uncertainty of misclassified observations.

3.3 Control Parameters

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

`tol` Iteration convergence tolerance. The default is `emControl()$tol=c(1.e-5,`$\sqrt{\epsilon_M}$`)`, where ϵ_M is the relative machine precision, which has the value `2.220446e-16` on IEEE compliant machines. 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"`, `"VEV"`).

`eps` A tolerance for terminating iterations due to ill-conditioning, such as near singularity in covariance matrices. The default is `emControl()$eps` which is set to the relative machine precision ϵ_M.

A function `emControl` is provided in `MCLUST` for setting these parameters and supplying default values. Although these control settings are in a sense hidden by the defaults, they may have a significant effect on results in some instances and should be taken into consideration in analysis.

4 Bayesian Information Criterion

`MCLUST` provides a function `bic` to compute the Bayesian Information Criterion (BIC) [26] given the maximized loglikelihood for model, the data dimensions, and the number of components in the model. 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 the `iris` dataset with the unconstrained variance model:

```
> meVVViris <- me(modelName = "VVV", data = iris[,-5], z = unmap(iris[,5]))

> bic( modelName = "VVV", loglik = meVVViris$loglik,
       n = nrow(iris), d = ncol(iris[,-5]), G = 3)
[1] -580.8397
```

5 Model-Based Hierarchical Clustering

`MCLUST` provides functions `hc` for model-based hierarchical agglomeration, and `hclass` for determining the resulting classifications. Function `hc` implements fast methods based on the multivariate normal classification likelihood [8]. We use the `iris` dataset distributed with `R` in our example. Figure 11 is a pairs plot of the `iris` dataset in which the three species are differentiated by symbol, obtained by the following command:

```
> clPairs(data = iris[,-5], classification = iris[,5])
```

Below we apply the hierarchical clustering algorithm for unconstrained covariances (`VVV`) to the `iris` dataset:

```
> hcVVViris <- hc(modelName = "VVV", data = iris[,-5])
```

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(hcVVViris, 2:3)
```

The classifications can be displayed with the data using `clPairs`:

```
> clPairs(data = iris[,-5], classification = cl[,"2"])
> clPairs(data = iris[,-5], classification = cl[,"3"])
```

More options for displaying clustering and classification results are discussed in Section 6. The 3-group classification can be compared with the known 3-group classification into species, which is given in the 5th column of the `iris` data, using function `classError`:

```
> classError(cl[,"3"], truth = iris[,5])
$misclassifiedPoints
 [1] 102 107 114 115 120 122 124 127 128 134 139 143 147 150

$errorRate
[1] 0.09333333
```

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.

Function `hc` for model-based hierarchical clustering based on the unconstrained (`VVV`) model is used to obtain the default initial values for the model-based clustering functions `Mclust` and `mclustBIC`.

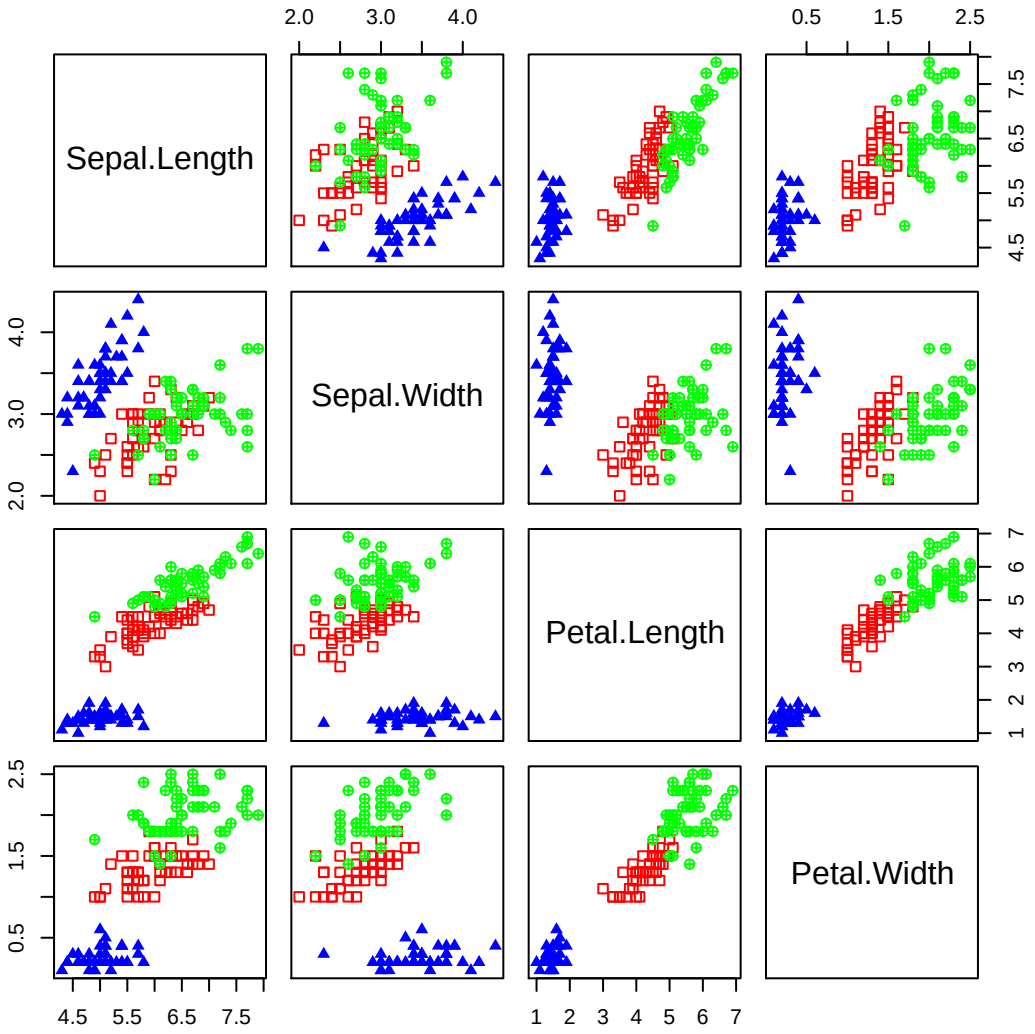

Figure 11: Pairs plot of the `iris` dataset showing classifcation into species.

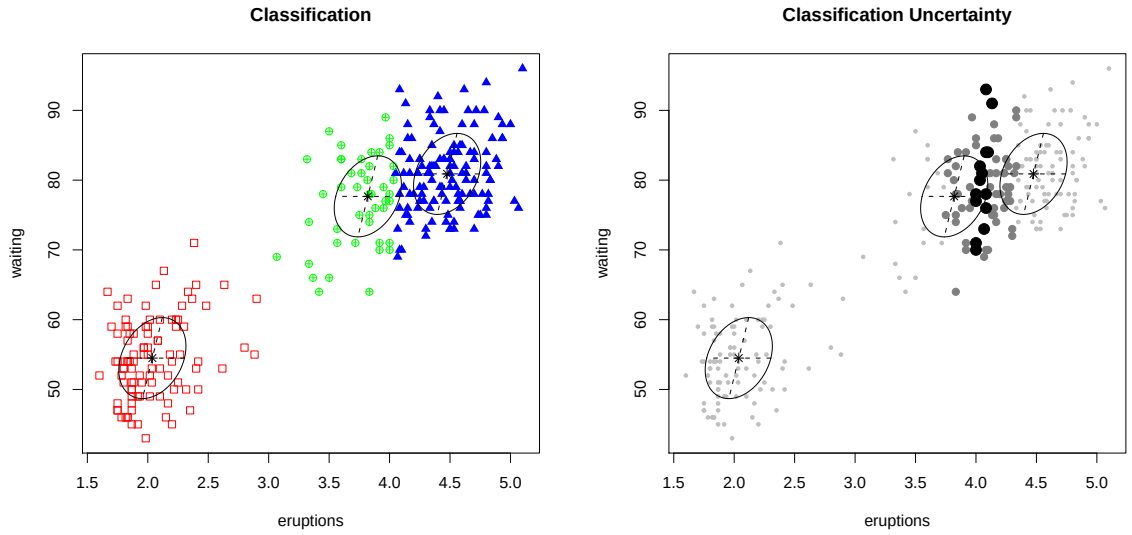

Figure 12: Classification (left) and uncertainty (right) plots created with `mclust2Dplot` for the `Mclust` model of the `faithful` dataset. The ellipses shown are the multivariate analogs of the standard deviations for each mixture component. In the classification plot, points in different classes are 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.

6 Displays for Multidimensional Data

Once parameter values of a mixture model fit are available, projections of the data showing the means and standard deviations of the corresponding components or clusters may be plotted. For bivariate data, density and uncertainty surfaces may also be plotted.

6.1 Displays for Bivariate Data

The function `mclust2Dplot` may be used for displaying the classification, uncertainty or classification errors for `MCLUST` models of bivariate data. In the following example, classification and uncertainty plots are produced for the `faithful` dataset in Figure 1.

```
> faithfulMclust <- Mclust(faithful)

> mclust2Dplot(data = faithful, what = "classification", identify = TRUE,
     parameters = faithfulMclust$parameters, z = faithfulMclust$z)

> mclust2Dplot(data = faithful, what = "uncertainty", identify = TRUE,
    parameters = faithfulMclust$parameters, z = faithfulMclust$z)
```

The resulting plots are displayed in Figure 12.

The function `surfacePlot` may be used for displaying the density or uncertainty for `MCLUST` models of bivariate data. It also returns the grid coordinates and corresponding

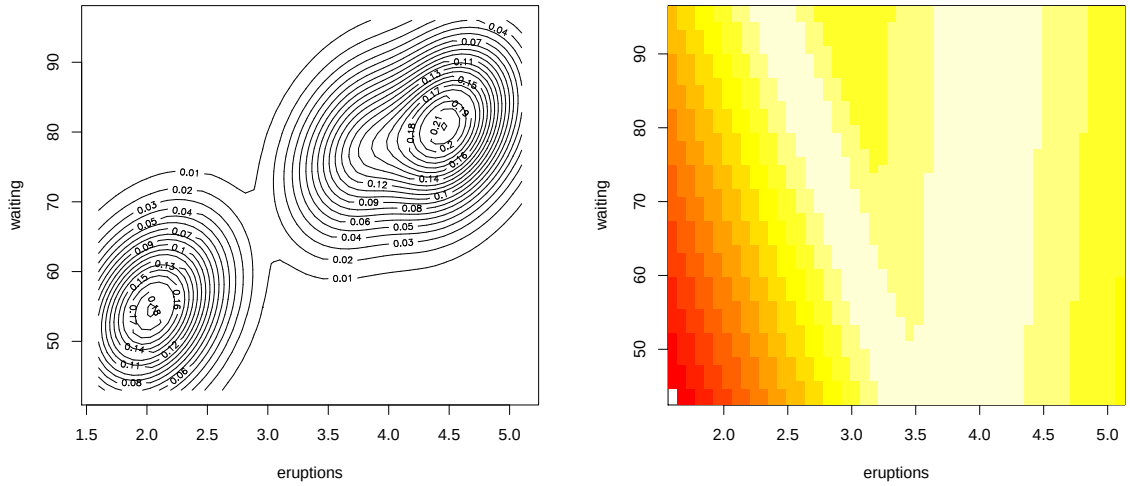

Figure 13: Density (left column) and uncertainty (right column) surfaces for the `faithful` dataset. A square root transformation was used for the density plot, which is plotted as a contour surface. A logarithmic transformation was used for the uncertainty plot, which is plotted as an image surface.

surface values. The following example shows how to display density and uncertainty surfaces for the `Mclust` model fit to the `faithful` dataset.

```
> surfacePlot(data = faithful, what = "density", type = "contour",
     parameters = faithfulMclust$parameters, transformation = "sqrt")

> surfacePlot(data = faithful, what = "uncertainty", type = "image",
    parameters = faithfulMclust$parameters,  transformation = "log")
```

The resulting plots are displayed in Figure 13.

6.2 Displays for Higher Dimensional Data

6.2.1 Coordinate Projections

To plot coordinate projections in MCLUST, use the function coordProj. The example we consider is a 3-group model for the iris dataset:

```
> irisBIC <- mclustBIC(iris[,-5])
> irisSummary3 <- summary(irisBIC, data = iris[,-5], G = 3)

> coordProj( data = iris[,-5], dimens = c(2,4), what = "classification",
   parameters = irisSummary3$parameters, z = irisSummary3$z)

> coordProj( data = iris[,-5], dimens = c(2,4), what = "uncertainty",
   parameters = irisSummary3$parameters, z = irisSummary3$z)

> coordProj( data = iris[,-5], dimens = c(2,4), what = "errors",
   parameters = irisSummary3$parameters, z = irisSummary3$z, truth = iris[,5])
```

These plots are displayed in Figure 14.

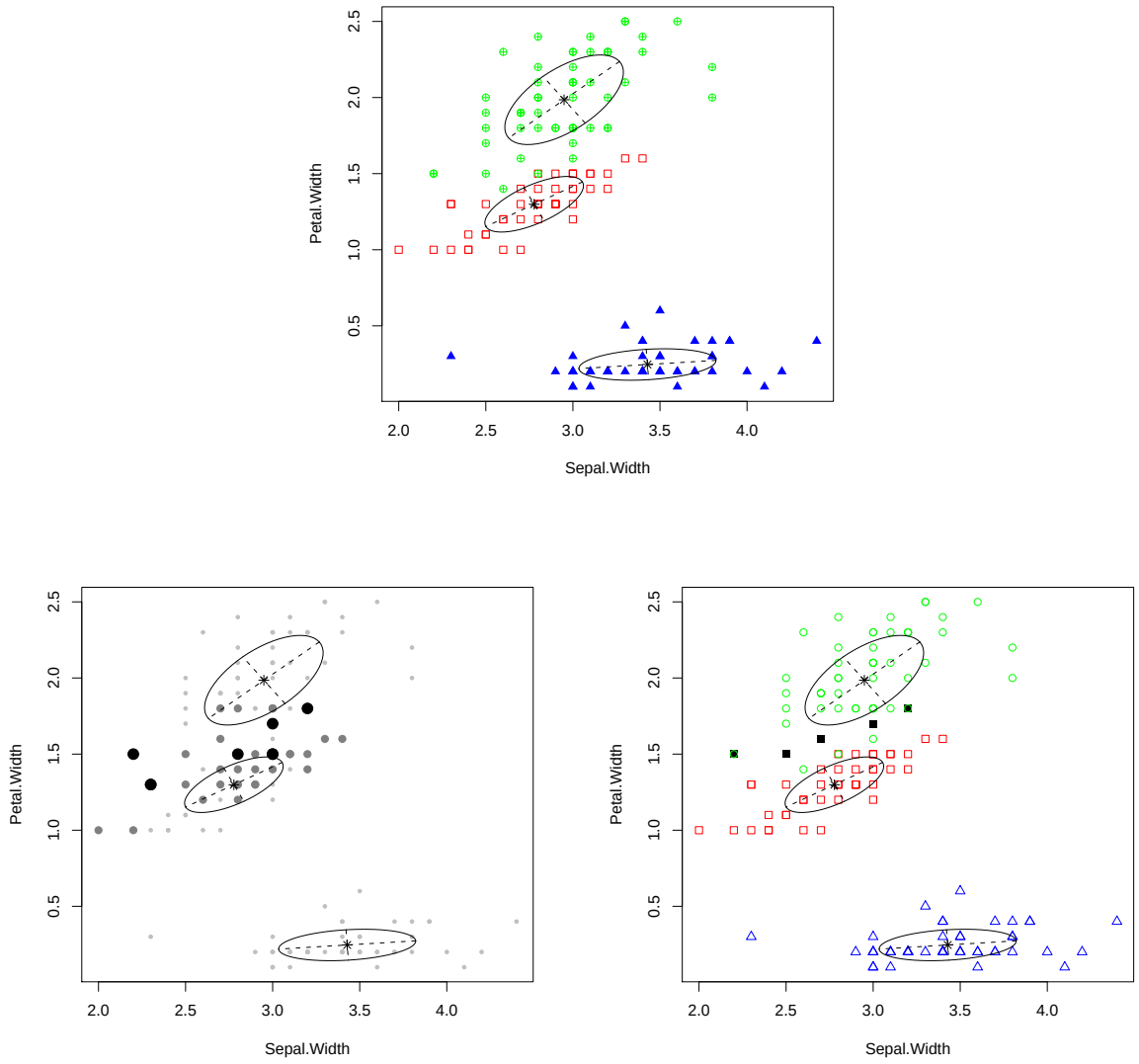

Figure 14: A coordinate projection of the `iris` dataset created with `coordProj`. Plots show the 3-group model-based classification (top) with associated uncertainty (bottom, left) and classification errors (bottom,right).

6.2.2 Random Projections

To plot random projections in `MCLUST`, use the function `randProj`. Again we consider is a 3-group model for the `iris` dataset:

```
> randProj( data = iris[,-5], seed = 43, what = "classification",
   parameters = irisSummary3$parameters, z = irisSummary3$z)

> randProj( data = iris[,-5], seed = 79, what = "classification",
   parameters = irisSummary3$parameters, z = irisSummary3$z)

> randProj( data = iris[,-5], seed = 201, what = "classification",
   parameters = irisSummary3$parameters, z = irisSummary3$z)
```

These plots are displayed in Figure 15.

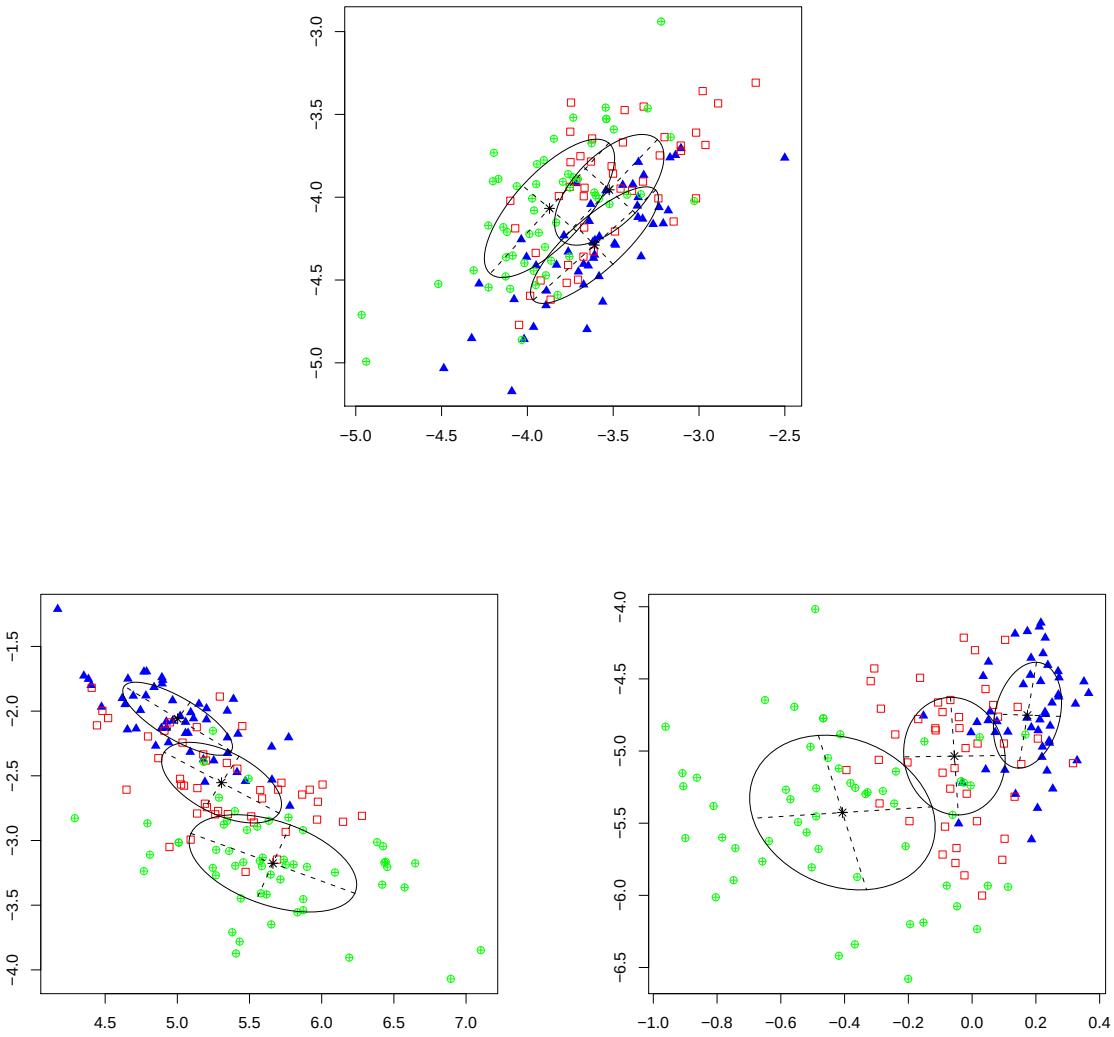

Figure 15: Some random projections of the `iris` dataset created with `randProj`. Plots show the 3-group classification from model-based clustering with three different seeds.

7 Density Estimation

The clustering capabilities of MCLUST can also be viewed as a general strategy for multivariate density estimation. After applying the clustering functions to fit a model to the data, function dens can be used to get the density of a given point relative to that model. As an example, we use the bivariate faithful dataset (see Figure 1).

First, we use Mclust or (mclustBIC and summary) to get a model for the data, as was done in Section 2:

```
> faithfulMclust <- Mclust(faithful)
```

The faithful dataset is bivariate, so for plotting the density can be computed over a grid. 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.

```
> apply(faithful, 2, range)
     eruptions waiting
[1,]       1.6      43
[2,]       5.1      96
> x <- grid1( 100, range = range(faithful$eruptions))
> y <- grid1( 100, range = range(faithful$waiting))
> xy <- grid2(x,y)
> xyDens <- dens(modelName = faithfulMclust$modelName, data = xy,
                 parameters = faithfulMclust$parameters)
> xyDens <- matrix(xyDens, nrow = length(x), ncol = length(y))
```

The result can be plotted using R functions contour, persp, or image since the dataset is bivariate.

```
> par(pty = "s")
> Z <- log(xyDens)
> persp(x = x, y = y, z = Z, box = FALSE)
> contour(x = x, y = y, z = Z, nlevels = 10)
> image(x = x, y = y, z = Z)
```

These plots are shown in Figure 16.

Probably the most common application for density estimation is discriminant analysis, for which a detailed discussion is given in Section 9.

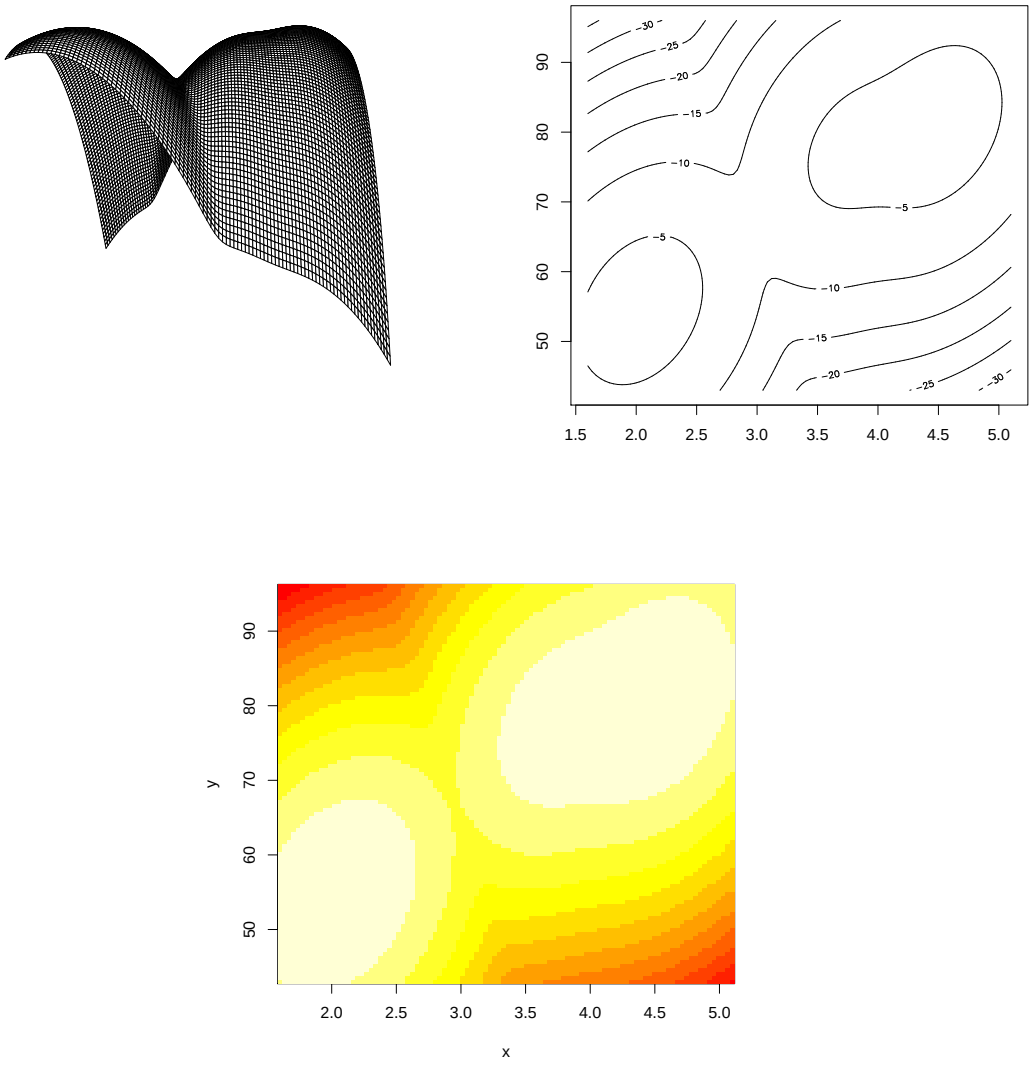

Figure 16: Perspective, contour and image plots of an `MCLUST` density estimate for the bivariate `faithful` dataset.

8 Simulation from Mixture Densities

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 two different datasets of the same size as the `faithful` dataset from the model produced by `Mclust` for the `faithful` dataset:

```
> faithfulMclust <- Mclust(faithful)

> sim0 <- sim( modelName = faithfulMclust$modelName,
               parameters = faithfulMclust$parameters,
               n = nrow(faithful), seed = 0)
> sim1 <- sim( modelName = faithfulMclust$modelName,
               parameters = faithfulMclust$parameters,
               n = nrow(faithful), seed = 1)
```

The results can be plotted as follows:

```
> xlim <- range(c(faithful[,1],sim0[,2],sim1[,2]))
> ylim <- range(c(faithful[,2],sim0[,3],sim1[,3]))

> mclust2Dplot(data=faithful, parameters = faithfulMclust$parameters,
  classification = faithfulMclust$classification, xlim = xlim, ylim = ylim)

> mclust2Dplot(data=sim0[,-1], parameters = faithfulMclust$parameters,
               classification = sim0[,1], xlim = xlim, ylim = ylim)

> mclust2Dplot(data=sim1[,-1], parameters = faithfulMclust$parameters,
               classification = sim1[,1], xlim = xlim, ylim = ylim)
```

The plots are shown in Figure 17. Note that `sim` produces a dataset in which the first column is the classification.

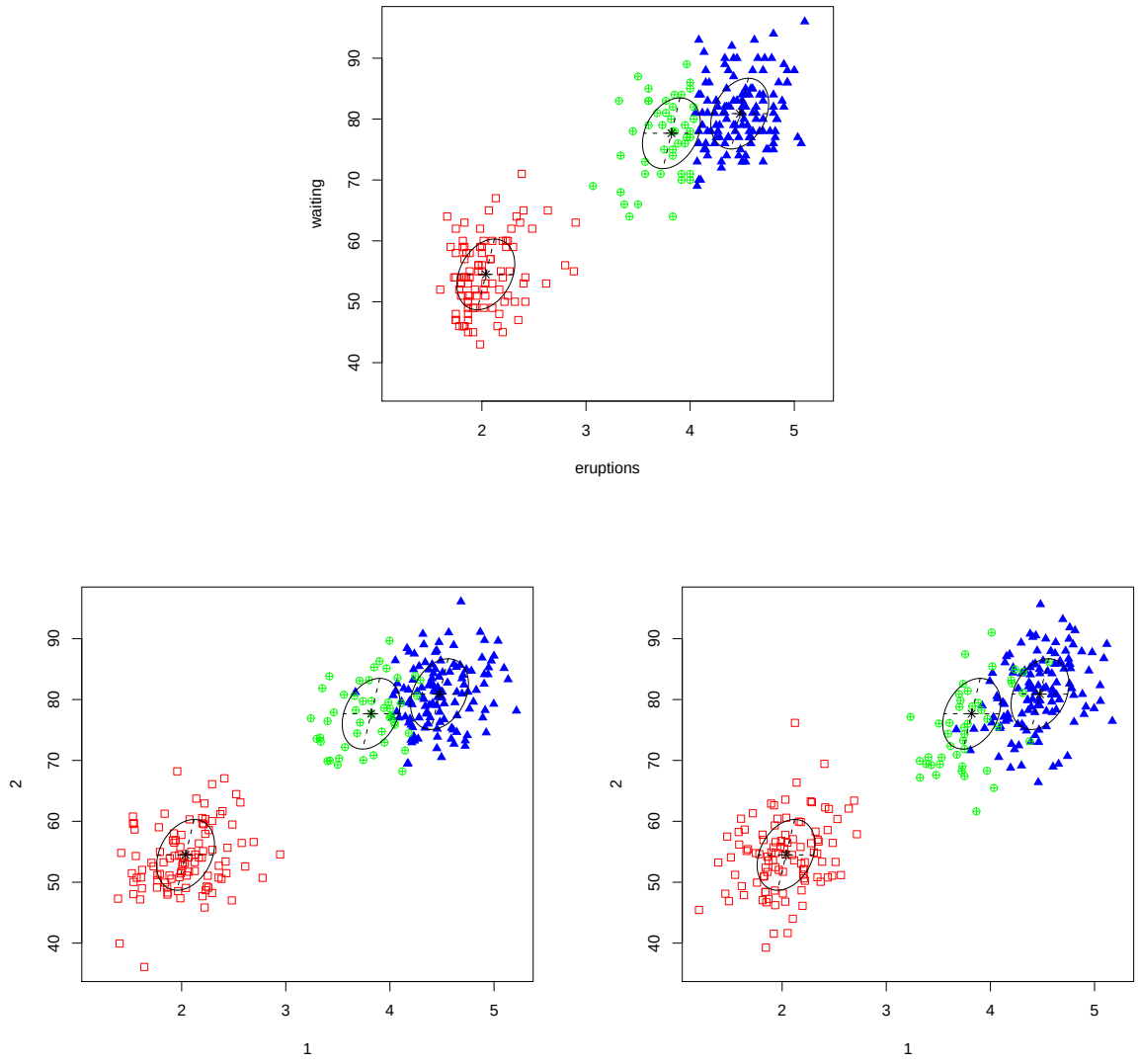

Figure 17: Data simulated from a model of the `faithful` dataset. The top hand figure is the `faithful` dataset, and the bottom figures are datasets of the same size simulated from the `Mclust` model for the `faithful` dataset. The ellipse defined by the covariance matrices of the model is shown on all of the plots.

9 Discriminant Analysis

In discriminant analysis, observations of known classification are used to classify others. MCLUST provides several approaches to discriminant analysis. We demonstrate some possible methods applied to the faithful dataset using the three-group model-based classification shown in Figure 2 as the ground truth:

```
> faithfulMclust <- Mclust(faithful)
> faithfulClass <- faithfulMclust$classification
```

9.1 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 cross validation. Function cv1EMtrain implements leave-one-out crossvalidation. Leaving out one training observation at a time, 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 faithful dataset as a training set, the result is:

```
> odd <- seq(from=1, to=nrow(faithful), by=2)
> round(cv1EMtrain(data = faithful[odd,], labels = faithfulClass[odd]),3)
  EII   VII   EEI   VEI   EVI   VVI   EEE   EEV   VEV   VVV
0.162 0.162 0.037 0.037 0.044 0.044 0.015 0.015 0.015 0.022
```

The crossvalidation error achieves a minimum for the elliptical, equal shape models EEE, EEV, and VEV. Of these we choose the most parsimonious model EEE. 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. [22]).

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

```
> modelEEE <- mstep(modelName = "EEE", data=faithful[odd,],
                    z=unmap(faithfulClass[odd]))
> classEEE <- map(estep(modelName = "EEE", data=faithful,
                        parameters = modelEEE$parameters)$z)
> classError(classEEE[odd], faithfulClass[odd])$errorRate
[1] 0.007352941

> even <- seq(from=2, to=nrow(faithful), by=2)
```

```
> classError(classEEE[even], faithfulClass[even])$errorRate
[1] 0.007352941
> classError(classEEE[even], faithfulClass[even])$misclassified
[1] 17
```

The error rates for the training [odd-numbered] data and the test [even-numbered] data are identical (.735%); two data points are misclassified:

```
> classError(classEEE, faithfulClass)$misclassified
[1] 34 71
```

The classification and the misclassified observations are shown in Figure 18. Not surprisingly, the misclassified observations fall in the region where the clusters overlap.

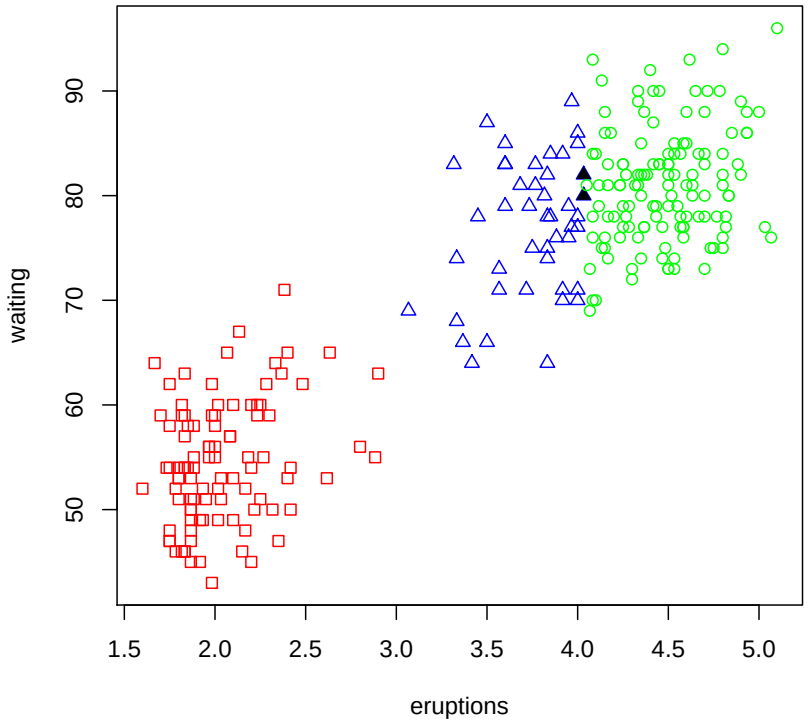

Figure 18: Classification errors from discriminant analysis for the `faithful` dataset using `mstep` and `estep`. Filled symbols are the misclassified data points.

Another option for model selection that is faster to compute than crossvalidation 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 within `MCLUST` for this purpose. For the `faithful` dataset, BIC for the models fitted to the odd-numbered observations is:

```
> round(bicEMtrain(faithful[odd,], labels = faithfulClass[odd]),0)
```

```
  EII   VII   EEI   VEI   EVI   VVI   EEE   EEV   VEV   VVV
-1761 -1771 -1187 -1196 -1194 -1204 -1183 -1193 -1202 -1210
```

BIC chooses `EEE` as the best model, so in this case the training and test errors are the same as for crossvalidation, and the classification results are as shown in Figure 18.

Although in this case crossvalidation and BIC happen to choose the same model, for other datasets the models selected, and hence the discriminant results, could be different. Cross-validation is much more computationally intensive procedure for model-selection than BIC, although timing comparisons between `cv1EMtrain` and `bicEMtrain` should not be considered a valid algorithmic comparison because there are more efficient ways to compute crossvalidation using updating schemes and compiled code.

9.2 Mixture Discriminant Analysis via MclustDA

In Section 9.1, discriminant analysis was accomplished by modeling the training data with a mixture density having 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. A more flexible alternative is to use model-based clustering to fit a Gaussian mixture model as a density estimate for each class in the training set. We illustrate the methods in this section with the 2-group model from model-based clustering for the `faithful` dataset as ground truth:

```
> faithfulBIC2 <- mclustBIC(faithful, G=2)
> faithfulClass2 <- summary(faithfulBIC2, faithful)$classification
```

9.2.1 `mclustDA`

If both training and test sets are given in advance, function `mclustDA` can be used for discriminant analysis. Its input is the training data and associated class labels, and the test data (optionally with labels). 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.

```
> faithfulMclustDA <- mclustDA(train = list(data = faithful[odd,],
                                    labels = faithfulClass2[odd]),
                               test = list(data = faithful[even,],
                                    labels = faithfulClass2[even]))
XXX EEI
  1   2
> faithfulMclustDA

Modeling Summary:
  trainClass mclustModel numGroups
1          1         XXX         1
2          2         EEI         2

Test Classification Summary:
```

```
  1   2
101  35

Training Classification Summary:

 1  2
75 61

Training Error: 0

Test Error: 0.007352941
```

The error rates for `mclustDA` classification are 0% and .735% for the training [odd-numbered] and test [even-numbered] data, respectively.

These discriminant analysis results can be plotted as follows:

```
> plot(faithfulMclustDA, trainData = faithful[odd,], testData = faithful[even,])
```

Figure 19 shows some plots of the results: The training models are shown in Figure 20.

9.2.2 `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 typically takes much more time than testing, it can be advantageous to separate training and testing computations. Function `mclustDAtrain` 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 9.1, except that in `mclustDA` a model for each class of the training data is chosen separately, instead of choosing a parameterized mixture model (which may have cross-cluster constraints) for all of the training data. `mclustDA` uses BIC (see section 4) to select the model. BIC adds a penalty term to the maximized loglikelihood that increases with the number of parameters.

By default, `mclustDAtrain` will fit up to three components for each possible model. Results for the odd-numbered observations in the `faithful` dataset are as follows:

```
> faithfulTrain <- mclustDAtrain(data = faithful[odd,],
                                 labels = faithfulClass2[odd])
XXX EEI
  1   2
```

The training models are shown in Figure 20.

The density of observations under the training models can be obtained using `mclustDAtest`, while the classification and posterior probabilities of the test or other data can be recovered

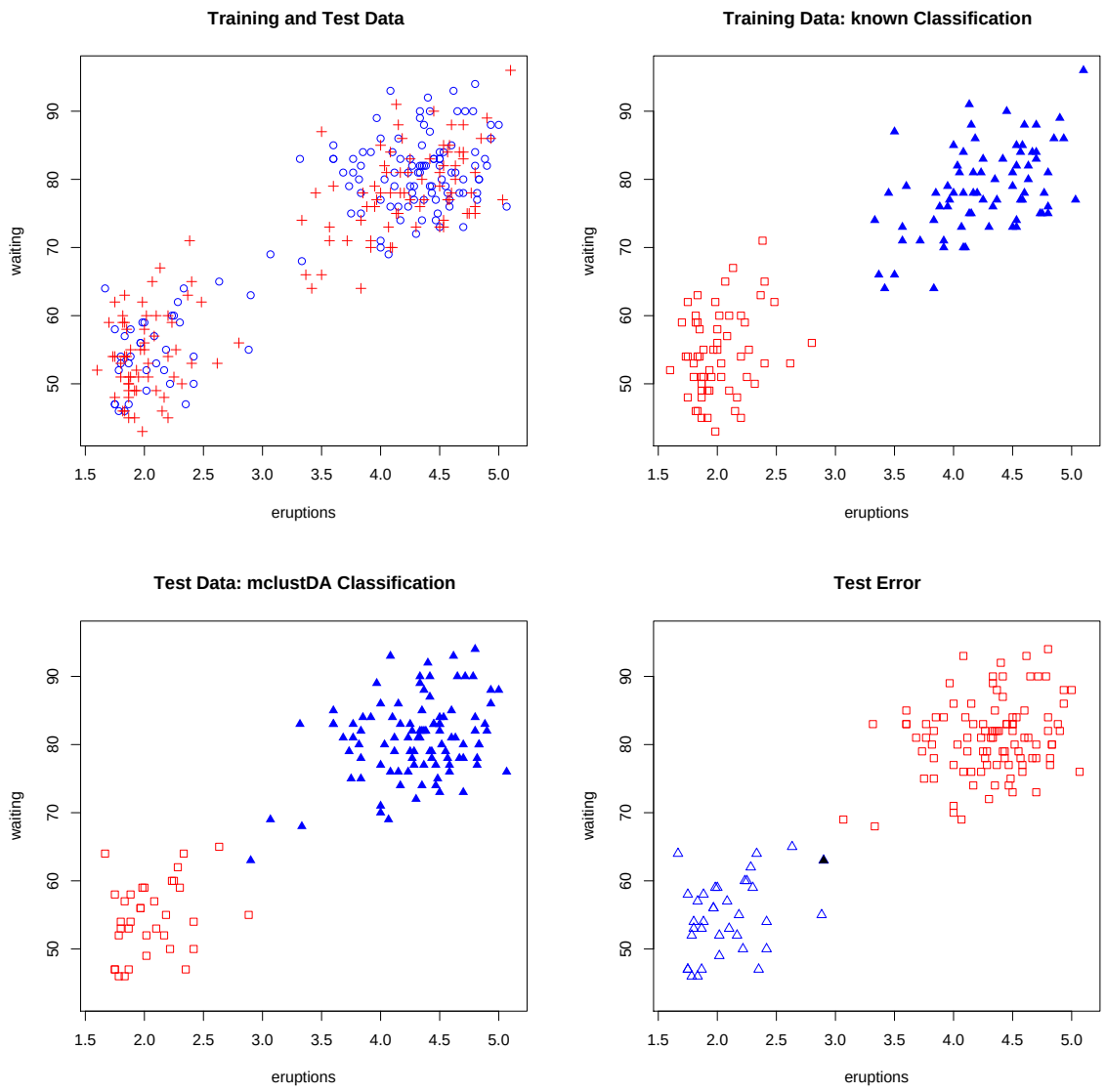

Figure 19: Plots associated with `mclustDA` on the `faithful` dataset. Upper Left: the training [odd-numbered/circles] and test [even-numbered/crosses] `faithful` data. Upper Right: the training data with known classification. Lower Left: the `mclustDA` classification of the test data. Lower Right: the errors (filled symbols) in using the `mclustDA` model to classify the test data.

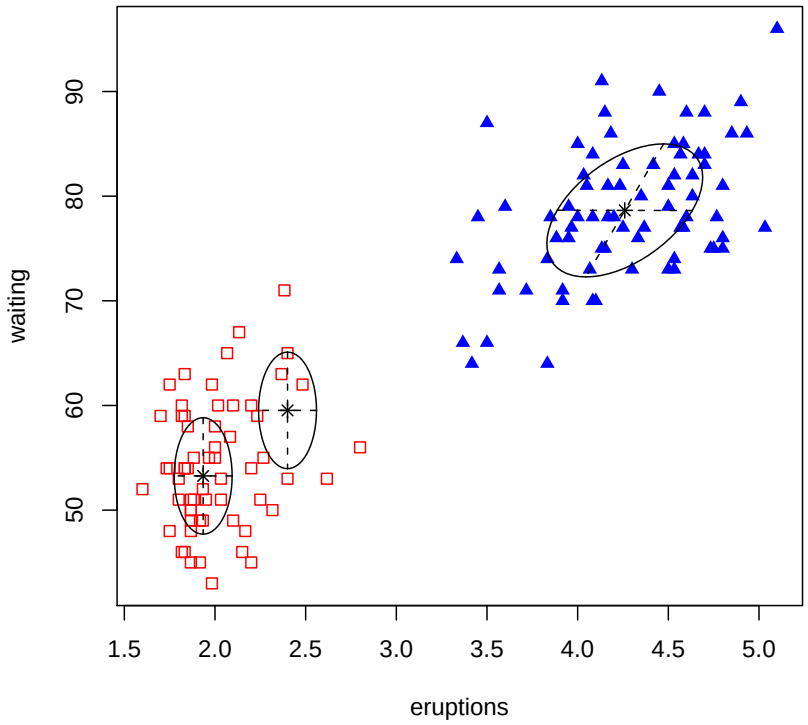

Figure 20: `mclustDA` training models for the odd numbered observations of the `faithful` dataset, using the two-group classification from model-based clustering as ground truth. One of the classes is modeled by a two-group equal variance diagonal model, and the other by a single unconstrained normal.

from the `summary` function for `mclustDAtest`. The test (even-numbered) classfication and error can be obtained as follows:

```
> faithfulEvenTest <- mclustDAtest(models=faithfulTrain, data=faithful[even,])

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

> classError(summary(faithfulEvenTest)$classification,
                     faithfulClass2[even])$errorRate

[1] 0.007352941
```

The training (odd-numbered) classfication and error can be obtained as follows:

```
> faithfulOddTest <- mclustDAtest(models=faithfulTrain, data=faithful[odd,])

> classError(summary(faithfulOddTest)$classification,
                     faithfulClass2[odd])$errorRate
[1] 0
```

10 Univariate Data

The MCLUST functions for clustering, density estimation and discriminant analysis can be applied to univariate 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).

10.1 Clustering

Cluster analysis for univariate data can be carried out as for two and higher dimensions. As an example, we use the precip dataset (included in the R language distribution):

```
> precipMclust <- Mclust(precip)

> plot(precipMclust, precip, legendArgs = list(x = "bottomleft"))
```

Figure 21 shows the BIC, classification, uncertainty, and density for this example.

The analysis can also be divided into two parts: BIC computation via mclustBIC and model computation via summary, as shown below for the rivers dataset (included in the R language distribution):

```
> riversBIC <- mclustBIC(rivers)
> plot(riversBIC)
> riversModel <- summary(riversBIC, rivers)
> riversModel

classification table:
 1  2  3
76 52 13

best BIC values:
      V,3       V,4       V,5
-2015.579 -2022.513 -2035.102
```

There is a special plotting function mclust1Dplot for univariate model-based clustering. As an example, we compare graphical results the 2-component maxmimum BIC model with the 3-component model:

```
> riversModel2 <- summary(riversBIC, rivers, G = 2)

> mclust1Dplot(data = rivers, what = "classification",
               parameters=riversModel$parameters, z=riversModel$z)
> mclust1Dplot(data = rivers, what = "density",
               parameters=riversModel$parameters, z=riversModel$z)
> abline(v = riversModel$parameters$mean, lty = 3)
```

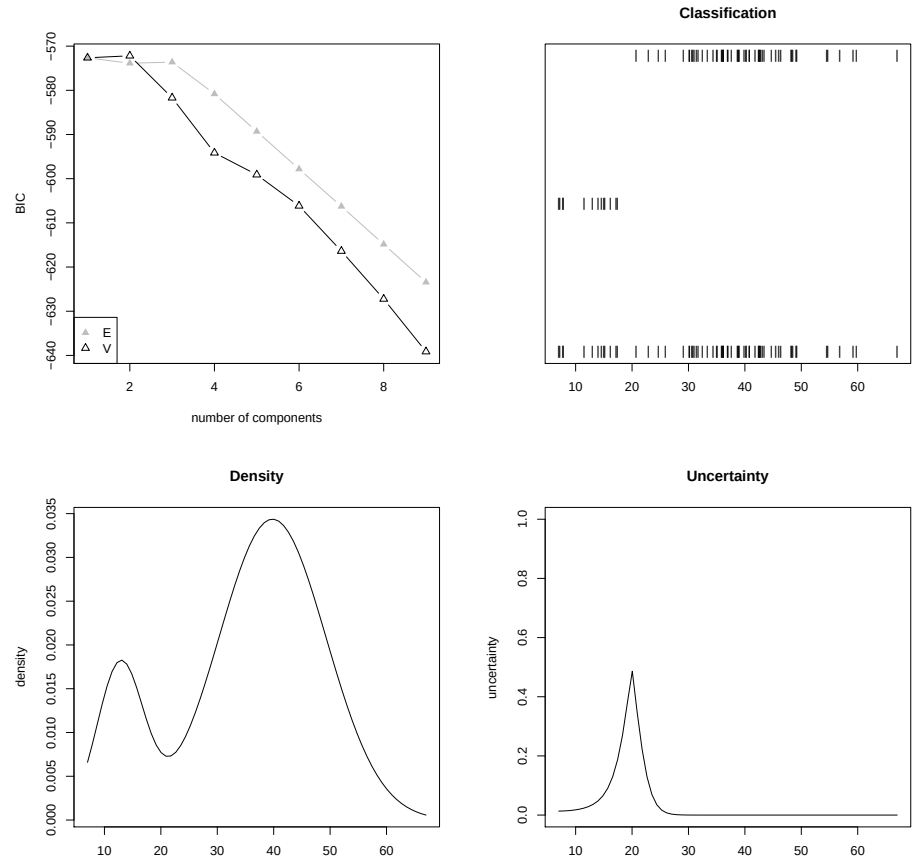

Figure 21: Model-based clustering of the univariate R dataset precip. Clockwise from upper left: BIC, classification, uncertainty, and density from Mclust applied to the simulated univariate example. In the classification plot, all of the data is displayed at the bottom, with the separated classes shown on different levels above.

```
> mclust1Dplot(data = rivers, what = "classification",
               parameters=riversModel2$parameters, z=riversModel2$z)
> mclust1Dplot(data = rivers, what = "density",
               parameters=riversModel2$parameters, z=riversModel2$z)
> abline(v = riversModel2$parameters$mean, lty = 3)
```

Vertical lines are added at the means of each component. Figure 22 shows the classification and density corresponding to the two- and three-component cases for this example. Density estimates can also be computed and plotted directly:

```
> points <- seq(from = min(rivers), to = max(rivers), length = 1000)
> riversDens3 <- dens(modelName = riversModel$modelName, data = points,
                      parameters = riversModel$parameters)
> plot(points, riversDens3, type = "l")
> abline(v = riversModel$parameters$mean, lty = 3)
```

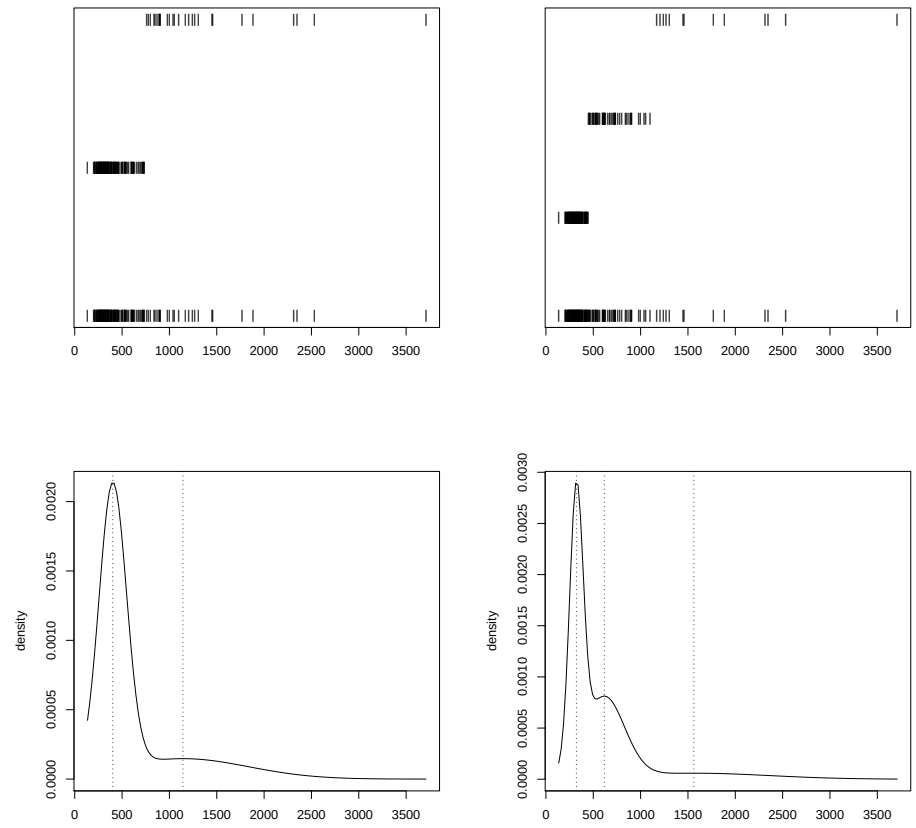

Figure 22: Two- and three-component classifications and densities for the univariate R dataset rivers. Vertical lines have been added to the density plots to show the location of the component means.

```
> riversDens2 <- dens(modelName = riversModel2$modelName, data = points,
                      parameters = riversModel2$parameters)
> plot(points, riversDens2, type = "l")
> abline(v = riversModel2$parameters$mean, lty = 3)
```

There is also a separate density function, densityMclust, that behaves similarly to the R function density.

```
> riversDens <- densityMclust(rivers)
> plot( riversDens, xlim = c(0,max(rivers)))
> plot( riversDens, data = rivers, xlim = c(0,max(rivers)))
```

The corresponding plots are shown in Figure 23. Data points are shown on a vertical line at the bottom of the plot when data is supplied to the plot function for densityMclust.

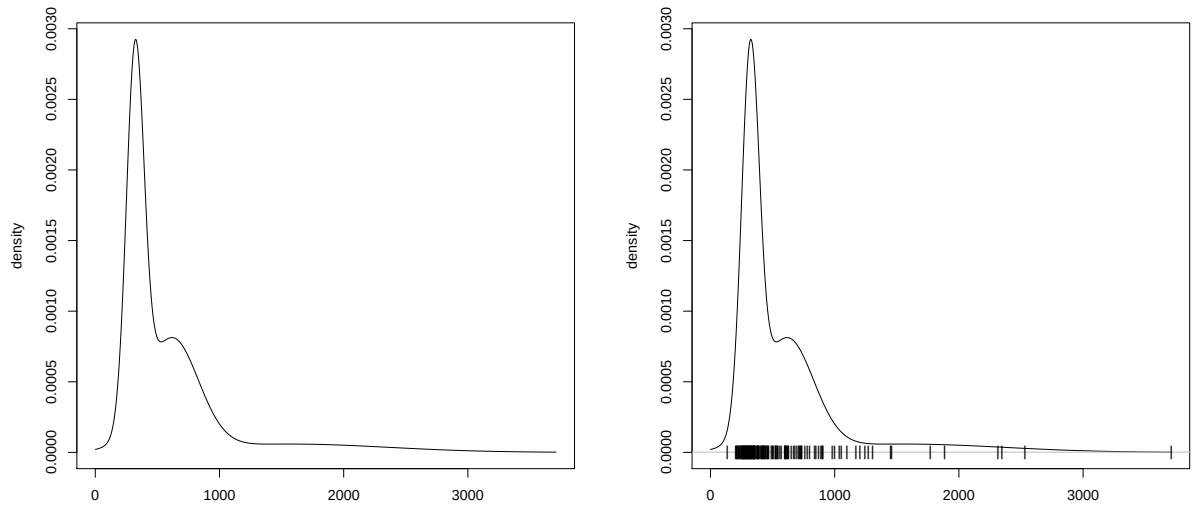

Figure 23: Plots of the density for the univariate `R` dataset `rivers` using function `densityMclust`. The right hand plot shows the augmented display when data points are supplied.

10.2 Discriminant Analysis

To illustrate discriminant analysis on univariate data, we use simulated data from a normal mixture consisting of two components with variance 1 centered at -9 and 9, respectively, and one component with variance 4 centered at 0:

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

We use the following simulated data as a test set:

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

The density of the distribution from which the training data is drawn is shown in Figure 24.

Discriminant Analysis via EM. In discriminant analysis via EM (Section 9.1), if we assume that each component constitutes a separate group,

```
> xClass <- c(rep(1,300),rep(2,400),rep(3,300))
> yClass <- c(rep(1,100),rep(2,100),rep(3,100))
```

then both leave-one-out crossvalidation and BIC choose the varying variance model `V` in the training stage:

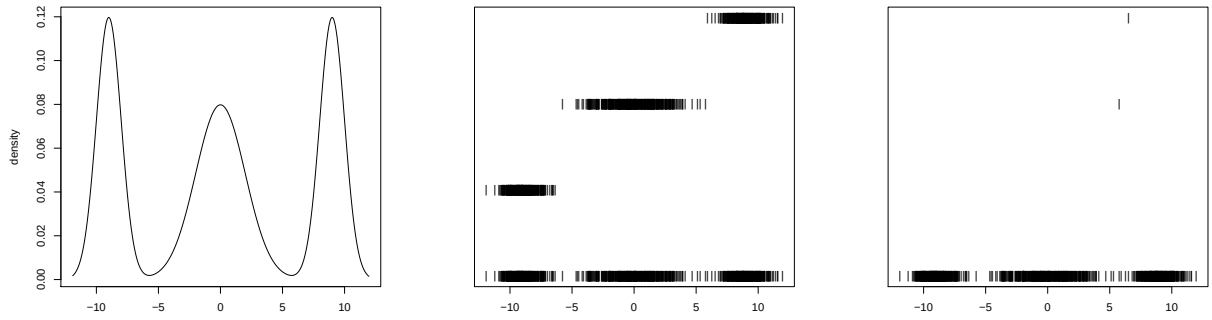

Figure 24: LEFT: Density for the univariate simulation data. There are two components with variance 1 centered at -9 and 9, respectively, and one component with variance 4 centered at 0. CENTER: Training step classification. RIGHT: Misclassified training observations.

```
> round(cv1EMtrain(x,labels=xClass),3)
    E     V
0.006 0.002
> round(bicEMtrain(x,labels=xClass),3)
        E         V
-5786.672 -5607.173
```

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

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

> classV <- map(estep(modelName = "V", data = c(x,y),
                      parameters = modelV$parameters)$z)

> classError(classV[1:length(x)],xClass)$errorRate ## training error
[1] 0.002

> classError(classV[1:length(x)],xClass)$misclassified
[1] 391 990

> classError(classV[-(1:length(x))],yClass)$errorRate ## test error
[1] 0
```

The classification and classification errors for the training data are shown in Figure 24.

Discriminant Analysis via MclustDA. To illustrate the discriminant analysis via the MclustDA methodology (Section 9.2): we used the same simultated univariate data but assume that observations are grouped by component variance:

```
> xClass <- c(rep(1,300),rep(2,400),rep(1,300))
> yClass <- c(rep(1,100),rep(2,100),rep(1,100))
```

The training stage fits a two component equal-variance model to one group, and a one-component model to the other:

```
> xTrain <- mclustDAtrain(x, labels = xClass)
 E X
 2 1
```

The classification error rates are the same as we obtained for discriminant analysis via EM with the 3-class grouping:

```
> xTest <- summary(mclustDAtest(x,xtrain))

> classError(xTest$classification,xClass)$errorRate ## training error
[1] 0.002
> classError(xTest$classification,xClass)$misclassified
[1] 391 990

> yTest <- summary(mclustDAtest(y,xTrain))

> classError(yTest$classification,yClass)$errorRate  ## testing error
[1] 0
```

The classification and classification errors for the training data are as shown in Figure 24.

11 Extensions

11.1 Large Datasets

`Mclust` and `mclustBIC` include a provision for using a subsample of the data in the hierarchical clustering phase before applying EM to the full data set, in order to extend the method to larger datasets. Some other methods for handling such cases are discussed in [30, 17]. The following example uses a random sample of size 100 in the initial hierarchical clustering phase of `EMclust` applied to the iris data:

```
> nrow(iris)
[1] 150
> S <- sample(1:nrow(iris), size = 100)
> Mclust(iris[,-5], initialization = list(subset = S))
```

For very large data sets, the discrimination capabilities of `MCLUST` can be used for classification. First, cluster analysis with the methodolgy of `Mclust`/`mclustBIC` 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 9.

11.2 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.

11.3 Missing Data

At present, `MCLUST` has no direct provision for handling missing values in data. However, a function `imputeData` has been added to the `MCLUST` package for creating datasets with missing data imputations using the `mix` package. Here we illustrate the use of the `imputeData` to fill in missing values in the continuous portion of the `stlouis` dataset provided with the `mix` package (we remove the first 3 (categorical) variables, since `MCLUST` is intended for continuous variables).

```
> library(mix)
```

```
> stlouis
     G D1 D2  R1  V1  R2  V2
4001 1 NA  1 110  NA  NA 150
4004 1  1  2 118 165  NA 130
4005 1  2 NA 116 145 114 125
.
.
.
3103 3 NA  2  NA  NA 100 140
3109 3  1  1 105 138  74  75
3111 3 NA NA  88 118  84 103

stlimp <- imputeData( stlouis[,-(1:3)])
```

Note that the values obtained for the missing entries will vary depending on the random number seed set in function `imputeData`, chosen randomly by default. It is usually desirable to combine multiple imputations in analyses involving missing data. See Little and Rubin (2002) for details and references on multiple imputation.

Another function `imputePairs` has been added to the `MCLUST` package for visualizing missing data imputations.

```
imputePairs( stlouis[,-(1:3)], stlimp[,-1])
```

A pairs plot showing the imputed values is displayed in Figure 25.

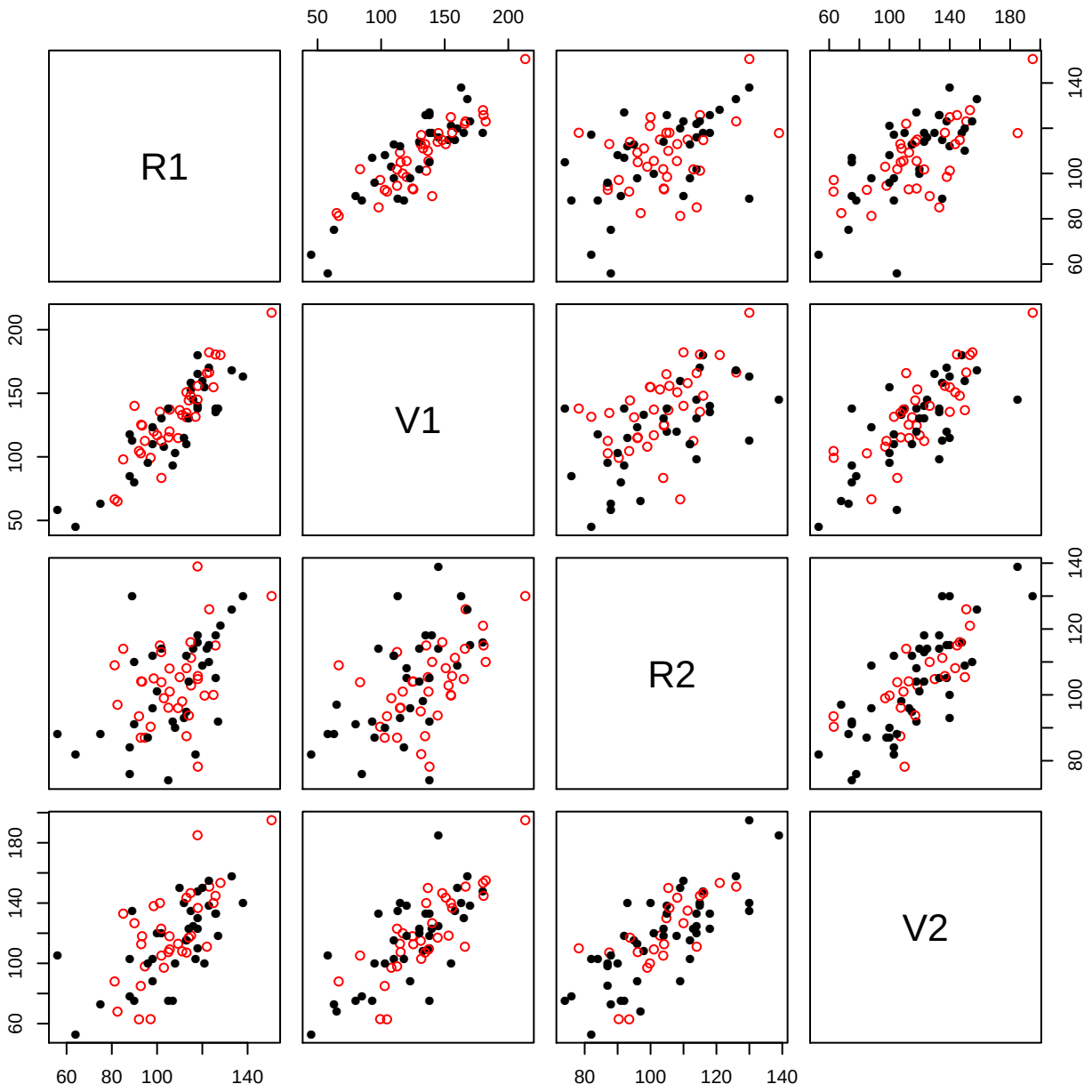

Figure 25: Example of missing data imputations obtained via the `mix` package for the continuous variables in the `stlouis` dataset. The closed black circles correspond to nonmissing data, while the open red circles correspond to imputed missing values.

12 Function Summary

12.1 Hierarchical Clustering

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

12.2 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.

12.3 Density Computation for Parameterized Gaussian Mixtures

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

12.4 Model-based Clustering / Density Estimation

`mclustBIC`	BIC computation; clusters and models through `summary`.
`Mclust`	Combines `mclustBIC` and its `summary` (fewer options).
`densityMclust`	Density estimation via `Mclust` (univariate).

12.5 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).

12.6 Bayesian Regularization

`priorControl`	specify a prior distribution.
`defaultPrior`	default prior distribution.

12.7 Support for Modeling and Classification

`.Mclust`	vector of default values.
`mclustOptions`	set `MCLUST` options.
`mclustModel`	specify an mclust model.
`mclustModelNames`	extract model names from an mclust model.
`mclustVariance`	extract (co)variance for an mclust model.
`emControl`	specify parameters affecting EM.
`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.
`mapClass`	Mapping between two classifications.
`classError`	Classification error.
`adjustedRandIndex`	Adjusted Rand Index.
`sigma2decomp`	Convert mixture covariances to decomposition form.
`decomp2sigma`	Convert decomposition form to mixture covariances.
`nVarParams`	Number of variance parameters.
`imputeData`	Missing data imputation using the `mix` package.

12.8 Plotting Functions

12.8.1 Univariate Data

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

`plot.densityMclust` `plot` method associated with the `densityMclust` function.

12.8.2 Bivariate Data

`mclust2Dplot` Classification, uncertainty, and/or classification errors.

`surfacePlot` Contour, image, or perspective plot of either density or uncertainty.

12.8.3 More than Two Dimensions

Classification, uncertainty, and/or classification errors.

`coordProj`	coordinate projections
`randProj`	random projections

12.8.4 Other Plotting Functions

`clPairs` pairs plot showing classification

`imputePairs` pairs plot to show missing data imputations

`uncerPlot` relative uncertainty of misclassified observations

`plot.Mclust` plots associated with `Mclust` results

`plot.mclustBIC` BIC plot associated with `mclustBIC` results

`plot.mclustDA` plots associated with `mclustDA` results

`plot.mclustDAtrain` plots associated with `mclustDAtrain` results

A Appendix: Clustering Models

`MCLUST` usually assumes a normal or Gaussian mixture model

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

where where $\mathbf{x}$ represents the data, G is the number of components, τ_k is the probabilty that an observation belongs to the kth component ($\tau_k \geq 0; \quad \sum_{k=1}^{G} \tau_k = 1$) , and

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

The exception is for model-based hierarchical clustering, for which the model used is the classification likelihood with a parameterized normal distribution assumed for each class:

$$\prod_{i=1}^{n} \phi_{\ell_i}(\mathbf{x}_i \mid \mu_{\ell_i}, \Sigma_{\ell_i}),$$

where the ℓ_i are labels indicating a unique classification of each observation: $\ell_i = k$ if $\mathbf{x}_i$ belongs to the kth component.

The components or clusters in both these models 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 (Banfield and Raftery 1993). 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 [24, 2, 6]. This parameterization includes but is not restricted to well-known variance models that are associated with various criterion for hierarchical clustering, such as equal-volume spherical variance ($\Sigma_k = \lambda I$) for the sum of squares criterion [29], constant variance [18], and unconstrained variance [27].

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. These are a subset of the parameterizations discussed in [6], which gives details of the EM algorithm for maximum likelihood estimation for these models.

A.1 Modeling Noise and Outliers

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

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

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, 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 Voronoï method [1], a nearest-neighbor method [3], and robust covariance estimation [28]. The function `NNclean` in the contributed `R` package `prabclus` is an implementation of the nearest-neighbor method. The function `cov.nnve` in the contributed `R` package `covRobust` is an implementation of robust covariance estimation. 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.

A.2 Model Selection via BIC

Several measures have been proposed for choosing the clustering model (parameterization and number of clusters); see, e.g., Chapter 6 of [23]. We use the Bayesian Information Criterion (BIC) approximation to the Bayes factor [26], which adds a penalty to the loglikelihood based on the number of parameters, and has performed well in a number of applications (e.g. [9, 11]). The BIC has the form

$$\text{BIC} \equiv 2 \ \text{loglik}_{\mathcal{M}}(\mathbf{x}, \theta_k^*) - (\#\ \text{params})_{\mathcal{M}} \ \log(n), \tag{3}$$

where $\text{loglik}_{\mathcal{M}}(\mathbf{x}, \theta_k^*)$ is the maximized loglikelihood for the model and data, $(\#\ \text{params})_{\mathcal{M}}$ is the number of independent parameters to be estimated in the model $\mathcal{M}$, and n is the number of observations in the data.

A.3 Adding a Prior to the Model

By default, `MCLUST` does not use a prior for modeling. However, users can optionally specify a conjugate prior of the type described in this section. For univariate data, we use a normal

prior on the mean (conditional on the variance):

$$\begin{aligned} \mu \mid \sigma^2 &\sim \mathcal{N}(\mu_{\mathcal{P}}, \sigma^2/\kappa_{\mathcal{P}}) \\ &\propto \left(\sigma^2\right)^{-\frac{1}{2}} \exp\left\{-\frac{\kappa_{\mathcal{P}}}{2\sigma^2}\left(\mu-\mu_{\mathcal{P}}\right)^2\right\} \end{aligned} \tag{4}$$

and an inverse gamma prior on the variance:

$$\begin{aligned} \sigma^2 &\sim \text{inverseGamma}(\nu_{\mathcal{P}}/2, \varsigma_{\mathcal{P}}^2/2) \\ &\propto \left(\sigma^2\right)^{-\frac{\nu_{\mathcal{P}}+2}{2}} \exp\left\{-\frac{\varsigma_{\mathcal{P}}^2}{2\sigma^2}\right\}. \end{aligned} \tag{5}$$

For multivariate data, we use a normal prior on the mean (conditional on the covariance matrix):

$$\begin{aligned} \mu \mid \Sigma &\sim \mathcal{N}(\mu_{\mathcal{P}}, \Sigma/\kappa_{\mathcal{P}}) \\ &\propto |\Sigma|^{-\frac{1}{2}} \exp\left\{-\frac{\kappa_{\mathcal{P}}}{2}\text{trace}\left[\left(\mu-\mu_{\mathcal{P}}\right)^T \Sigma^{-1}\left(\mu-\mu_{\mathcal{P}}\right)\right]\right\}, \end{aligned} \tag{6}$$

and an inverse Wishart prior on the covariance matrix:

$$\begin{aligned} \Sigma &\sim \text{inverseWishart}(\nu_{\mathcal{P}}, \Lambda_{\mathcal{P}}) \\ &\propto |\Sigma|^{-\frac{\nu_{\mathcal{P}}+d+1}{2}} \exp\left\{-\frac{1}{2}\text{trace}\left[\Sigma^{-1}\Lambda_{\mathcal{P}}^{-1}\right]\right\}. \end{aligned} \tag{7}$$

The hyperparameters $\mu_{\mathcal{P}}$, $\kappa_{\mathcal{P}}$, and $\nu_{\mathcal{P}}$ are called the *mean*, *shrinkage* and *degrees of freedom*, respectively. Parameters $\varsigma_{\mathcal{P}}^2$ (a scalar) and $\Lambda_{\mathcal{P}}$ (a matrix) are the *scale* of the prior distribution in the univariate and multivariate cases, respectively. These priors are called *conjugate priors* for the normal distribution because the posterior can be expressed as the product of a normal distribution and an inverse gamma or Wishart distribution. With the prior, a modified version of the BIC, in which the MLE is replaced by the MAP, is used to choose the number of clusters. Details on model-based clustering with a prior can be found in [15].

Functions `priorControl` and `defaultPrior` are provided in `MCLUST` for specifying a prior. When called with defaults, the following choices are made for the prior hyperparameters:

$\mu_{\mathcal{P}}$: the mean of the data.

$\kappa_{\mathcal{P}}$: .01
The posterior mean $\dfrac{n_k\bar{y}_k + \kappa_{\mathcal{P}}\mu_{\mathcal{P}}}{\kappa_{\mathcal{P}} + n_k}$ can be viewed as adding $\kappa_{\mathcal{P}}$ observations with value $\mu_{\mathcal{P}}$ to each group in the data. The value we used was determined by experimentation; values close to and bigger than 1 caused large perturbations in the modeling in cases where there were no missing BIC values without the prior. The value .01 resulted in BIC curves that appeared to be smooth extensions of their counterparts without the prior.

$\nu_{\mathcal{P}}$: $d+2$
Analogously to the univariate case, the marginal prior distribution of μ is a t distribution centered at $\mu_{\mathcal{P}}$ with $\nu_{\mathcal{P}} - d + 1$ degrees of freedom. The mean of this distribution

is $\mu_{\mathcal{P}}$ provided that $\nu_{\mathcal{P}} > d$, and it has a finite covariance matrix provided $\nu_{\mathcal{P}} > d + 1$ (see, e. g. Schafer 1997). We chose the smallest integer value for the degrees of freedom that gives a finite covariance matrix.

$\varsigma^2_{\mathcal{P}}$: $\frac{\text{sum}(\text{diag}(\text{var}(\text{data})))/d}{G^{2/d}}$ (For univariate models, and multivariate spherical or diagonal models.) The average of the diagonal elements of the empirical covariance matrix of the data divided by the square of the number of components to the $1/d$ power. This is roughly equivalent to partitioning the range of the data into G intervals of fairly equal size.

$\Lambda_{\mathcal{P}}$: $\frac{\text{var}(\text{data})}{G^{2/d}}$ (For multivariate ellipsoidal models.) The empirical covariance matrix of the data divided by the square of the number of components to the $1/d$ power.

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.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] C. Fraley. Algorithms for model-based Gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20:270–281, 1998.

[9] C. Fraley and A. E. Raftery. How many clusters? Which clustering method? - Answers via model-based cluster analysis. *The Computer Journal*, 41:578–588, 1998.

[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.

[12] C. Fraley and A. E. Raftery. Enhanced software for model-based clustering, density estimation, and discriminant analysis: MCLUST. *Journal of Classification*, 20:263–286, 2003.

[13] C. Fraley and A. E. Raftery. Model-based microarray image analysis. *R News*, 6:60–63, 2006.

[14] C. Fraley and A. E. Raftery. Some applications of model-based clustering in chemistry. *R News*, 6:17–23, 2006.

[15] C. Fraley and A. E. Raftery. Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of Classification*, 24:155–181, 2007.

[16] C. Fraley and A. E. Raftery. Model-based methods of classification: using the `mclust` software in chemometrics. *Journal of Statistical Software*, 18(6), January 2007.

[17] C. Fraley, A. E. Raftery, and R. Wehrens. Incremental model-based clustering for large datasets with small clusters. *Journal of Computational and Graphical Statistics*, 14:1–18, 2005.

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

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

[20] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[21] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley, 2nd edition, 2002.

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

[23] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

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

[25] J. L. Schafer. *Analysis of Incomplete Multivariate Data by Simulation*. Chapman and Hall, 1997.

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

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

[28] N. Wang and A. E. Raftery. Nearest neighbor variance estimation (NNVE): Robust covariance estimation via nearest neighbor cleaning (with discussion). *Journal of the American Statistical Association*, 97:994–1019, 2002.

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

[30] R. Wehrens, L. Buydens, C. Fraley, and A. Raftery. Model-based clustering for image segmentation and large datasets via sampling. *Journal of Classification*, 21:231–253, 2004.