COMBINED ALGORITHMS FOR FITTING FINITE MIXTURE DISTRIBUTIONS

COMBINED ALGORITHMS FOR CONSTRAINED ESTIMATION OF FINITE MIXTURE DISTRIBUTIONS WITH GROUPED DATA AND CONDITIONAL DATA

By

JUAN DU, B.Sc.

A Project

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (2002) **McMaster University**

(Statistics) **Hamilton, Ontario**

TITLE: Combined Algorithms for Constrained Estimation of Finite Mixture Distributions with Grouped Data and Conditional Data

AUTHOR: Juan Du, B.Sc. (Nankai University)

SUPERVISOR: Dr. Peter D.M. Macdonald, B.Sc., M.Sc. (Toronto University), D.Phil. (Oxford University)

NUMBER OF PAGES: xii, 124

Dedicated to

the Loving Memory

of

My Parents

Lianyou Du & Ming Wu

Abstract

Finite mixture distributions have been used as models throughout the history of modern statistics. A mixture distribution is a compounding of statistical distributions, which arises when sampling from inhomogeneous populations with a different probability density function in each component. A finite mixture has a finite number of components.

The objective of this project is to develop statistical software for estimating the parameters of mixture distributions with grouped data and conditional data. The method we apply is the standard maximum likelihood estimation method. However, estimating the parameters of a mixture distribution is difficult when the components are heavily overlapped. Macdonald & Pitcher (1979) solved the problem of overparameterization by applying constraints on parameters.

We develop a package called *Rmix* for the *R* statistical computing environment to fit finite mixture distributions, with the functionality of Macdonald's MIX software (1979), but with updated and substan-

tially improved numerical methods based on a combination of the EM algorithm and a Newton-type method implemented in the function `nlm` provided by the *R* system. A number of utility functions were also written to aid in preparing data and starting values for the parameters, plotting mixture distributions, providing the analysis of variance, and so on.

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Peter D.M. Macdonald for being not only an excellent guide but also a true mentor. Thank you very much for your continued guidance, support, encouragement and patience.

A special note of gratitude is due to the department of Mathematics and Statistics at McMaster University, particularly to my committee members Dr. Macdonald, Dr. Fox and Dr. Viveros for their helpful suggestions and advice.

I also wish to express my heartfelt thanks to my parents, my grandma, my brother Ye Wu and my boyfriend Yong Yang, for their encouragement and moral support especially through the valleys of my life. Lastly, I am very grateful to my friends Hong Qian, Tao Li, Sharon and James for their help and best wishes.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Finite Mixture Distribution

A mixture distribution is a compounding of statistical distributions, which arises when sampling from inhomogeneous populations (or mixed populations) with a different probability density function in each component. Size-frequency distributions in animal populations with distinct age-groups, the distribution of times to failure in a mixture of good and defective items, and the distribution of some diagnostic measure in a mixed population of patients, some of whom have a given disease and some of whom do not, are all examples of mixed distributions. A finite mixture has a finite number of components.

Definition 1.1.1 *Suppose that a random variable X takes values in a sample space $\mathfrak{X}$, and that its distribution can be represented by a prob-*

ability density function (or mass function in the case of discrete $\mathfrak{X}$) of the form

$$g(x) = \pi_1 f_1(x) + \cdots + \pi_k f_k(x) \quad (x \in \mathfrak{X}), \tag{1.1}$$

where

$$0 \leq \pi_i \leq 1, \quad i = 1, \ldots, k; \quad \pi_1 + \cdots + \pi_k = 1.$$

We say that X has a finite mixture distribution and that $g(\cdot)$ is a ***finite mixture density function****. The parameters $\pi_1, \ldots, \pi_k$ will be called the* ***mixing weights*** *or* ***mixing proportions*** *and $f_1(\cdot), \ldots, f_k(\cdot)$ the* ***component densities*** *of the mixture.*

There is no requirement that the component densities should all belong to the same parametric family, but throughout this project, we will restrict attention to the simplest case, where $f_1(x), \ldots, f_k(x)$ have a common functional form but have different parameters. We can then write $f_i(x) = f(x|\,\boldsymbol{\theta}_i)$ where $\boldsymbol{\theta}_i$ denotes the parameters occurring in $f_i(x)$. The finite mixture density function will then have the form

$$g(x|\,\boldsymbol{\Psi}) = \sum_{i=1}^{k} \pi_i f(x|\,\boldsymbol{\theta}_i) \quad (x \in \mathfrak{X}), \tag{1.2}$$

where $\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_k, \boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_k^T)^T$ is the complete collection of all distinct parameters occurring in the mixture model.

With many direct applications, there are situations where we believe in the existence of k underlying *categories*, or *groups*, such that the experimental unit on which the observation X is made belongs to one of these categories. We do not, however, observe directly the source of X. In these forms of application, $f(\cdot|\,\boldsymbol{\theta}_i)$ summarizes the probability distribution of X given that the observation actually derives from group i, and π_i denotes the probability that the observation comes from this source.

To fit finite mixture distributions, we need to estimate all the parameters in the mixture models. There is a remarkable variety of estimation methods that have been applied to finite mixture problems, such as the method of moments, maximum likelihood, minimum chi-square, least squares approaches and so on. We shall consider the most well-known maximum likelihood estimation method and take the likelihood function as our starting point. The likelihood functions have different forms with different kinds of data, we will give a brief introduction of complete and incomplete data with their likelihood functions in next sections.

1.2 Complete Data Problem

We say that a sample from a mixed population is complete if, for every observation, both the measurement X and the component the individual

come from are observed.

Definition 1.2.1 *Suppose that a random sample $x_1, x_2, \ldots, x_n$ arises from a mixed population, then* ***complete data*** *can be defined* ([TSM85]) *as*

$$\{y_j\,; j = 1, 2, \ldots, n\} = \{(x_j, \boldsymbol{z}_j);\ j = 1, 2, \ldots, n\},$$

where each $\boldsymbol{z}_j = (z_{1j}, \ldots, z_{kj})$ is an indicator vector of length k with 1 *in the position corresponding to the appropriate category and zeros elsewhere.*

For example, in studies of fish population we have a sample ([MP79]) of fish lengths from 523 pikes of *known* age. The five components correspond to groups of fish aged one to five, all older fish having been eliminated from the sample. Here $x_1, x_2, \ldots, x_{523}$ denote the lengths of pike, and for each x_j, z_{ij} will be equal to 1 in the position corresponding to the ith age group to which the fish belongs and zeros elsewhere. Table 1.1 gives part of the data.

The likelihood function corresponding to the complete data $(y_1, \ldots, y_n)$ can then be written in the form

$$L(\boldsymbol{\Psi};\, y_1, \ldots, y_n) = \prod_{j=1}^{n} g(x_j|\,\boldsymbol{\Psi}) = \prod_{j=1}^{n} \left[\prod_{i=1}^{k} \pi_i^{z_{ij}} f(x_j|\,\boldsymbol{\theta}_i)^{z_{ij}} \right] \qquad (1.3)$$

Table 1.1: Pike lengths and age

Num	Length	Age					Num	Length	Age				
		1	2	3	4	5			1	2	3	4	5
1	18.0	1	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	19.0	1	0	0	0	0	454	53.0	0	0	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	455	44.5	0	0	0	1	0
55	28.5	1	0	0	0	0	456	44.5	0	0	0	1	0
56	25.0	0	1	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	⋮
57	25.5	0	1	0	0	0	501	64.5	0	0	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	502	49.5	0	0	0	0	1
298	41.5	0	1	0	0	0	503	52.0	0	0	0	0	1
299	31.0	0	0	1	0	0	⋮	⋮	⋮	⋮	⋮	⋮	⋮
300	32.0	0	0	1	0	0	523	76.5	0	0	0	0	1

with logarithm

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{n} \sum_{i=1}^{k} z_{ij} \log \pi_i + \sum_{j=1}^{n} \sum_{i=1}^{k} z_{ij} \log f(x_j | \, \boldsymbol{\theta}_i) \,, \tag{1.4}$$

where π_i and $f(\cdot | \, \boldsymbol{\theta}_i)$ represent the mixing proportion and component density function for the ith component, respectively. Complete data estimation is straightforward when there are no constraints on the parameters: each component is fitted separately and the proportions are estimated by counting the number of observations in each component. Even in some situations where there are constraints imposed on the parameters, the likelihood function for complete data also has explicit

solutions.

1.3 Incomplete Data Problem

In many practical applications, the data sets are not complete, so we concentrate attention on the more difficult incomplete-data situations, particularly the grouped observations problem where the components are not observed but only the marginal distribution of X is available, and the conditional data problem where the components are observed at given values of X. The likelihood function for these incomplete data problems is given as

$$L(\boldsymbol{\Psi}) = \prod_{j=1}^{n} g(x_j|\,\boldsymbol{\Psi}) = \prod_{j=1}^{n} \left[\sum_{i=1}^{k} \pi_i f(x_j|\,\boldsymbol{\theta}_i) \right]. \tag{1.5}$$

From the formula above, we can see that incomplete data result in a complicated likelihood function and make it difficult to obtain maximum likelihood estimates.

1.3.1 Grouped Data

Grouped data are the observations grouped into intervals, where individual observations are not recorded but only the class intervals in which they fall, and the number of observations falling in each interval are recorded. Grouped observations are often given in the form of a

histogram.

Definition 1.3.1 *Let X be a random variable with p.d.f. $f(x|\boldsymbol{\Psi})$ specified up to a vector $\boldsymbol{\Psi}$ of unknown parameters. Suppose that the sample space $\mathfrak{X}$ of X is partitioned into m mutually exclusive intervals $\mathfrak{X}_j\,(j = 1, \ldots, m)$ with cell (or bin) boundaries $a_0, \ldots, a_m$, such that the jth cell corresponds to the interval $[a_{j-1}, a_j)$. Independent observations are made on $\mathfrak{X}$, but only the frequency $n_j\,(j = 1, \ldots, m)$ which denotes the number of observations falling in the jth bin is recorded. The data having the form as follows*

$$\{y_j; j = 1, 2, \ldots, m\} = \{(a_j, n_j);\ j = 1, 2, \ldots, m\},$$

*are referred to as **grouped data**. Usually, a_0 is 0 or negative infinity, so we don't display it.*

Size-frequency data are the most common grouped data. Samples of fish and other animals have been presented as size-frequency distributions for a long time because the appearance of separate modes may be interpreted as revealing age-groups. For the sample of the 523 pike lengths, we group the data over size, i.e. *length*, intervals to get length-frequency data with 25 bins and display it in Table 1.2.

For given

$$n = \sum_{j=1}^{m} n_j\,,$$

Table 1.2: Pike lengths: grouped data

Bin	Length	Frequency	Bin	Length	Frequency	Bin	Length	Frequency
1	19.75	4	10	37.75	44	19	55.75	8
2	21.75	10	11	39.75	42	20	57.75	3
3	23.75	21	12	41.75	36	21	59.75	6
4	25.75	11	13	43.75	23	22	61.75	6
5	27.75	14	14	45.75	22	23	63.75	3
6	29.75	31	15	47.75	17	24	65.75	2
7	31.75	39	16	49.75	12	25	Inf	5
8	33.75	70	17	51.75	12			
9	35.75	71	18	53.75	11			

the observed grouped data

$$\boldsymbol{y} = (n_1, \ldots, n_m)^T$$

has a multinomial distribution, consisting of n draws on m categories with probabilities $P_j(\boldsymbol{\Psi})\,(j = 1, \ldots, m)$. Thus the likelihood function for the grouped data $\boldsymbol{y}$ is given by

$$L(\boldsymbol{\Psi}) = \frac{n!}{n_1! \cdots n_m!} \{P_1(\boldsymbol{\Psi})\}^{n_1} \cdots \{P_m(\boldsymbol{\Psi})\}^{n_m} \tag{1.6}$$

and the log-likelihood is

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{m} n_j \log P_j(\boldsymbol{\Psi}) + c, \tag{1.7}$$

where

$$c = \log\left\{ n! \Big/ \prod_{j=1}^{m} n_j! \right\}$$

does not depend on $\boldsymbol{\Psi}$,

$$P_j(\boldsymbol{\Psi}) = \int_{a_{j-1}}^{a_j} f(x|\,\boldsymbol{\Psi})\, dx \quad j = 1, \ldots, m,$$

is the theoretical probability that an individual x belongs to the jth interval and $f(x|\,\boldsymbol{\Psi})$ the p.d.f. of X.

1.3.2 Conditional Data

Conditional data are based on grouped data from mixed populations. In some applications, in addition to the grouped data, there are also subsamples available of observations drawn from the selected bins to determine their individual class-memberships. The subsamples are called *conditional data*. Since conditional data bring additional information to the grouped data problem, the estimates of parameters are usually better.

Definition 1.3.2 *Suppose that $y_1, \ldots, y_m$ are grouped data; we select h bins and let $n_{jr}\,(r = 1, \ldots, h)$ denote the frequency of the $\{jr\}$th bin. For each selected bin, we draw a subsample of size $c_{jr}\,(0 < c_{jr} \leq n_{jr})$ to determine which groups they belong to, then count the number of observations falling into each group; the vectors consisting of the numbers*

are called ***conditional data*** *and are presented in the form*

$$\{\boldsymbol{c}_r = (c_{1,jr}\,, \ldots, c_{k,jr}), \quad r = 1, \ldots, h\}$$

and

$$0 \le c_{i,jr} \le c_{jr}\,, \quad \sum_{i=1}^{k} c_{i,jr} = c_{jr},$$

where $c_{i,jr}$ *denotes the number of observations from the* $\{jr\}$th *bin that belong to the* ith *group.*

For the lengths of pikes again, we select the 4, 5, 12, 13, 15–22nd bins, draw 11, 14, 36, 21, 10, 12, 12, 11, 8, 3, 6, 6 observations as subsamples, respectively, and then determine the age of these fish by biological methods. The conditional data obtained are presented in Table 1.3.

For grouped data from a mixture distribution with p.d.f. (1.2), the log likelihood is

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{m} n_j \log \left\{ \sum_{i=1}^{k} \pi_i P_{j|i}(\boldsymbol{\theta}_i) \right\}, \tag{1.8}$$

where

$$P_{j|i}(\boldsymbol{\theta}_i) = \int_{a_{j-1}}^{a_j} f(x|\,\boldsymbol{\theta}_i)\, dx \quad i = 1, \ldots, k\,;\ j = 1, \ldots, m,$$

denotes the probability of an individual from the ith component falling into the jth bin and $f(x|\,\boldsymbol{\theta}_i)$ is the component density of the ith group. To improve the resolution of highly overlapping component distributions, we draw subsamples from some bins to determine the components

Table 1.3: Pike lengths: conditional data

Bin	Length	Frequency	Age					Bin	Length	Frequency	Age				
			1	2	3	4	5				1	2	3	4	5
1	19.75	4						14	45.75	22					
2	21.75	10						15	47.75	17	0	0	5	5	0
3	23.75	21						16	49.75	12	0	0	6	5	1
4	25.75	11	9	2	0	0	0	17	51.75	12	0	0	5	7	0
5	27.75	14	8	6	0	0	0	18	53.75	11	0	0	4	4	3
6	29.75	31						19	55.75	8	0	0	0	8	0
7	31.75	39						20	57.75	3	0	0	0	2	1
8	33.75	70						21	59.75	6	0	0	0	1	5
9	35.75	71						22	61.75	6	0	0	0	2	4
10	37.75	44						23	63.75	3					
11	39.75	42						24	65.75	2					
12	41.75	36	0	2	34	0	0	25	Inf	5					
13	43.75	23	0	0	21	0	0								

to which they belong. For each selected bin, the conditional data $\boldsymbol{c}_r$ has a multinomial distribution, consisting of c_{jr} draws on k categories with the conditional probabilities $P_{i|jr}(\boldsymbol{\Psi})\,(i = 1, \ldots, k)$. Thus the log likelihood function for the grouped data with conditional data is

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{m} n_j \log \left\{ \sum_{i=1}^{k} \pi_i P_{j|i}(\boldsymbol{\theta}_i) \right\} + \sum_{jr, r=1,\ldots,h} \sum_{i=1}^{k} c_{i,jr} \log \{P_{i|jr}(\boldsymbol{\Psi})\} \tag{1.9}$$

where

$$P_{i|jr}(\boldsymbol{\Psi}) = \frac{\pi_i P_{jr|i}(\boldsymbol{\theta}_i)}{\sum_{i'=1}^{k} \pi_{i'} P_{jr|i'}(\boldsymbol{\theta}_i)}, \quad r = 1, \ldots, h, \tag{1.10}$$

denotes the probability that a subsampled unit from the $\{jr\}$th bin belongs to the ith group.

1.4 Objective of the Project

The objective of this project is to develop statistical software for fitting finite normal, log-normal, gamma, Weibull, binomial, negative binomial and Poisson mixture distributions with constrained parameters to grouped data and conditional data.

For estimating the mixing proportions and the parameters of the component distributions, we apply the standard maximum likelihood estimation (MLE) method, but a problem of over-parameterization exists. Macdonald & Pitcher (1979) solved this problem by applying constraints on the parameters. In Chapter 2, the constraints to be imposed on the parameter spaces are illustrated in some commonly occurring situations such as the proportions fixed, the means equal, the coefficients of variation fixed, and others.

Chapter 3 will introduce the EM algorithm and Newton-type methods, present their nature and explain the reason why we choose these algorithms and combine them in a particular way.

In Chapter 4, we will describe the computer programs in the *Rmix* library, written to assist users in estimating the parameters of a mixture distribution with grouped data and conditional data. A brief introduction of design principles and functionality is given, and the estimation procedure using the combination of the EM algorithm and a Newton-type method will be illustrated.

In Chapter 5, examples using *Rmix* to estimate the parameters of mixture distributions are provided.

Difficult cases arise in some specific environments, and Chapter 6 will discuss some strategies to avoid those problems. Remarks for the algorithms employed in this project and suggestions for future research will be also presented.

Chapter 2

Constraints on the Parameters

Fitting a mixture to data needs to estimate the mixing proportions π_i and the parameters $\boldsymbol{\theta}_i\,(i = 1, \ldots, k)$ of the component distributions, but, for theoretical and practical reasons it will not always be possible to estimate all of the parameters, particularly when the components overlap and obscure one another. Thus it is often desirable to reduce the number of parameters to be estimated. Macdonald and Pitcher (1979) approached this problem by assuming constraints.

Not only are the constraints discussed by Macdonald and Green (1998) considered, we also develop some new constraints for the situations where the new relations among parameters are applied. We assume that the mixture is composed of two-parameter components and, without loss of generality, the parameters are taken to be the mean and standard deviation. Then the parameters to be constrained are the

mixing proportions π_i, means μ_i, and standard deviations σ_i. Various combinations of the parameter constraints will also be applied. In the unconstrained case (no constraints imposed) we will estimate all the parameters in the model.

2.1 Constraints on Proportions

For the mixing proportions, the natural constraint $\pi_1 + \pi_2 + \cdots + \pi_k = 1$, where $0 \leq \pi_i \leq 1\,(i = 1, \ldots, k)$, is always imposed.

2.1.1 Proportions Free (NONE)

Only the natural constraint is applied. All the proportions but the last one will be estimated. The last one can be computed from the relation

$$\pi_k = 1 - \pi_1 - \cdots - \pi_{k-1}\,. \tag{2.1}$$

The abbreviation "NONE" is the code in the package *Rmix* that denotes that none of the constraints is imposed on the proportions.

2.1.2 Specified Proportions Fixed (PFX)

In addition to the natural constraint, any or all of the proportions may be held fixed while other proportions are being estimated. If a is the number of proportions fixed, the number of free proportions is $k - a - 1$, where

the -1 comes from the natural constraint. To constrain the proportions to be equal, hold each one fixed at $1/k$. We specify this constraint by the code "PFX" which means that some proportions are fixed. Each constraint presented in this chapter has its own code in *Rmix*.

2.2 Constraints on Means

2.2.1 Means Free (NONE)

Under this constraint, we need to estimate all the means $\mu_1, \ldots, \mu_k$.

2.2.2 Specified Means Fixed (MFX)

Specified means are held at fixed values while the remaining means are being estimated.

2.2.3 Means Equal (MEQ)

This constraint assumes that $\mu_1 = \mu_2 = \cdots = \mu_k$. We attempt to estimate their common value, and the common value is initialized at μ_1. This constraint is allowed if there are at least two components and the standard deviations are all different from each other; such a mixture is called a "scale mixture".

2.2.4 Equally Spaced Means (MES)

We apply this constraint in the situation where we assume that $(\mu_2 - \mu_1) = (\mu_3 - \mu_2) = \cdots = (\mu_k - \mu_{k-1})$. Only two means, μ_1 and μ_2, are estimated directly. Subsequent means are computed from the relation

$$\mu_i = \mu_1 + (i-1)(\mu_2 - \mu_1), \quad i = 3, \ldots, k. \tag{2.2}$$

This constraint is possible if there are at least three components.

2.2.5 Growth Curve (MGC)

This constraint forces the means to lie along a von Bertalanffy growth curve of the form

$$\mu_i = L_\infty\{1 - \exp[-\kappa(t_i - t_0)]\} \tag{2.3}$$

where

$$L_\infty = \mu_1 + \frac{(\mu_2 - \mu_1)^2}{(\mu_2 - \mu_1) - (\mu_3 - \mu_2)} \tag{2.4}$$

$$\kappa = -\log\left(\frac{\mu_3 - \mu_2}{\mu_2 - \mu_1}\right) \tag{2.5}$$

$$(t_i - t_0) = -\kappa^{-1}\log\left(1 - \frac{\mu_1}{L_\infty}\right). \tag{2.6}$$

Interpretation of the above parameters in the context of fisheries application is given by Macdonald and Green (1988): for components (age-groups) spaced exactly one year apart, μ_i is the mean fish size in the ith

age-group (age in years), t_0 is the hypothetical age at zero size, t_i is the actual age of the ith age-group, L_∞ is the hypothetical ultimate mean size in the population and κ is the growth parameter.

We only need to estimate the first three means μ_1, μ_2, μ_3. The remaining ones can be computed from the formula

$$\mu_i = \mu_1 + \frac{(\mu_2 - \mu_1)^2}{(\mu_2 - \mu_1) - (\mu_3 - \mu_2)} \left\{ 1 - \left(\frac{\mu_3 - \mu_2}{\mu_2 - \mu_1} \right)^{i-1} \right\}, \quad i = 4, \ldots, k. \tag{2.7}$$

The growth curve constraint cannot be used unless there are four or more components and $(\mu_3 - \mu_2) < (\mu_2 - \mu_1)$.

2.3 Constraints on Standard Deviations

The component standard deviations $\sigma_1, \ldots, \sigma_k$ are referred to as "sigmas" here.

2.3.1 Sigmas Free (NONE)

We will attempt to estimate all k standard deviations $\sigma_1, \ldots, \sigma_k$. If all the proportions and all the means are also being estimated, this choice is not likely to work unless the k components show as k clear modes in the histogram.

2.3.2 Specified Sigmas Fixed (SFX)

Specified standard deviations will be held fixed while we attempt to estimate the remaining sigmas.

2.3.3 Sigmas Equal (SEQ)

This constraint assumes that $\sigma_1 = \sigma_2 = \cdots = \sigma_k$. We attempt to estimate their common value and initialize it at σ_1. If there are two or more components and the means are all different from each other, then it is possible to employ this constraint.

2.3.4 Fixed Coefficients of Variation (FCV)

For this constraint, the coefficients of variation $(\sigma_1/\mu_1) = (\sigma_2/\mu_2) = \cdots = (\sigma_k/\mu_k)$ will all be held at the same fixed positive value c, and thus the means completely determine the standard deviations. We do not count the standard deviations as estimated parameters, and they can be computed from the relation

$$\sigma_i = c\,\mu_i\,, \quad i = 1, \ldots, k. \tag{2.8}$$

This constraint is permitted if all of the means are positive and different from each other.

Note that if the components are gamma distributions, fixing the coef-

ficients of variation at 1 will force them to be exponential distributions, since, for the gamma distribution, $\sigma/\mu = p^{-1/2}$, where p is the shape parameter ([R65], p. 133), and a gamma distribution with $p = 1$ is an exponential distribution.

2.3.5 Constant Coefficient of Variation (CCV)

This constraint assumes that $(\sigma_1/\mu_1) = (\sigma_2/\mu_2) = \cdots = (\sigma_k/\mu_k) = c$, where c is a parameter to be estimated rather than a fixed value, and is initialized at (σ_1/μ_1). We estimate σ_1 and then compute the other standard deviations from the formula

$$\sigma_i = c\,\mu_i = \frac{\sigma_1}{\mu_1}\,\mu_i\,, \quad i = 2, \ldots, k. \tag{2.9}$$

In addition to the condition of acceptance mentioned in the last constraint, this constraint also requires at least two components.

2.3.6 Binomial Relation (BINOM)

This constraint assumes that the relation between mean and standard deviation is

$$\sigma_i = \sqrt{\mu_i - \frac{{\mu_i}^2}{n_i}}\,, \quad i = 1, \ldots, k. \tag{2.10}$$

Since the binomial distribution has such a relation between the mean and standard deviation, we call this constraint the *binomial relation*,

where n_i is the number of trials for each component. We don't estimate sigmas but just compute them from the above formula.

2.3.7 Negative Binomial Relation (NBINOM)

We term the relation between mean and standard deviation, which has the form

$$\sigma_i = \sqrt{\mu_i + \frac{{\mu_i}^2}{n_i}} \ , \quad i = 1, \ldots, k, \tag{2.11}$$

where n_i is the number of trials for each component, a *negative binomial relation*, since the negative binomial distribution has such a relation between the mean and standard deviation. Under this constraint, we estimate the means only.

2.3.8 Poisson Relation (POIS)

If the relation between mean and standard deviation is of the form

$$\sigma_i = \sqrt{\mu_i} \ , \quad i = 1, \ldots, k, \tag{2.12}$$

we refer to it as *Poisson relation* for a reason similar to the last two constraints.

2.4 Combinations of the Constraints

Combinations of the constraints on the proportions, means and standard deviations may be applied according to various practical situations, but not all of the combinations are allowed. We shall give a brief discussion of situations where some combinations of constraints are restricted.

First of all, the equal means cannot have equal standard deviations, or the components of the mixture distribution will be reduced. Hence, the MEQ can't, obviously, have SEQ. According to this rule, some other combinations of the constraints on the means and standard deviations are also not employed. For example, if the constraint MEQ is applied on the means, the sigmas computed from one of (2.8), (2.9), (2.12) will be equal, so the combinations of MEQ and FCV, CCV, POIS are not allowed.

Secondly, for the given data and initial values of the parameters, some constraints are invalid. If the data or the initial values of the means have negative values, we can't use the constraints FCV, CCV and POIS.

Besides, when the distribution of components is specified, only the appropriate constraints can be applied. For instance, it is impossible to use NBINOM and POIS for binomial distribution components, since the variances have to be less than the corresponding means. Similarly, negative binomial distributions may not have the BINOM and POIS

constraints.

In brief, various situations should be considered according to theoretical and practical restrictions before choosing constraints.

Chapter 3

Numerical Algorithms

3.1 Maximum Likelihood Estimation

For estimating the unknown parameters, we apply the standard maximum likelihood estimation (MLE) method. Not only is it appealing on intuitive grounds, but it also possesses desirable statistical properties such as, under very general conditions, the estimates obtained by the method are consistent (they converge with probability 1 to the true parameter values). We first establish a general likelihood function and give the likelihood equation. Further, a brief introduction of the information matrix and score statistic will be presented.

Suppose X is a random variable or vector with probability density function (p.d.f.) $f(x|\boldsymbol{\Psi})$, where $\boldsymbol{\Psi} = (\Psi_1, \ldots, \Psi_d)^T$ is the parameter vector we wish to estimate. The parameter space is denoted by $\boldsymbol{\Omega}$.

Although we are taking X to be a continuous random variable, we can still view $f(x|\boldsymbol{\Psi})$ as a p.d.f. in the case where X is discrete by the adoption of counting measure.

If $x_1, \ldots, x_n$ denotes an observed independent random sample of size n on the random variable X, then

$$\boldsymbol{x} = (x_1, \ldots, x_n)^T.$$

The likelihood function for $\boldsymbol{\Psi}$ formed from the observed data $\boldsymbol{x}$ is given by

$$L(\boldsymbol{\Psi}; \boldsymbol{x}) = \prod_{j=1}^{n} f(x_j|\boldsymbol{\Psi}). \tag{3.1}$$

We attempt to find the particular $\boldsymbol{\Psi}$ that maximizes the likelihood function. This maximization can be dealt with in the traditional way by differentiating $L(\boldsymbol{\Psi}; \boldsymbol{x})$ with respect to the components of $\boldsymbol{\Psi}$ and equating the derivatives to zero to give the likelihood equation

$$\partial L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} = \mathbf{0}, \tag{3.2}$$

or equivalently,

$$\partial \log L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} = \mathbf{0}. \tag{3.3}$$

We let

$$\boldsymbol{I}(\boldsymbol{\Psi}; \boldsymbol{x}) = -\partial^2 \log L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} \partial \boldsymbol{\Psi}^T \tag{3.4}$$

be the matrix of the negative of the second-order partial derivatives of the log likelihood function with respect to the elements of $\boldsymbol{\Psi}$. Under

regularity conditions, the expected (Fisher) information matrix $\boldsymbol{\mathcal{I}}(\boldsymbol{\Psi})$ is given by

$$\begin{aligned} \boldsymbol{\mathcal{I}}(\boldsymbol{\Psi}) &= E\ \{\boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi})\boldsymbol{S}^T(\boldsymbol{x}|\boldsymbol{\Psi})\} \\ &= -E\ \{\boldsymbol{I}(\boldsymbol{\Psi};\boldsymbol{x})\}, \end{aligned} \tag{3.5}$$

where

$$\boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi}) = \partial \log L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} \tag{3.6}$$

is the gradient vector of the log likelihood function; that is, the score statistic. Here the operator E denotes expectation using the parameter vector $\boldsymbol{\Psi}$.

The asymptotic covariance matrix of the MLE $\hat{\boldsymbol{\Psi}}$ is equal to the inverse of the expected information matrix $\boldsymbol{\mathcal{I}}(\boldsymbol{\Psi})$, which can be approximated by $\boldsymbol{\mathcal{I}}(\hat{\boldsymbol{\Psi}})$; that is, the standard error of $\hat{\Psi}_i = (\hat{\boldsymbol{\Psi}})_i$ is given by

$$SE(\hat{\Psi}_i) \approx (\boldsymbol{\mathcal{I}}^{-1}(\hat{\boldsymbol{\Psi}}))_{ii}^{1/2}, \quad i = 1, \ldots, d, \tag{3.7}$$

where the standard notation $(\boldsymbol{A})_{ij}$ is used for the element in the ith row and jth column of a matrix $\boldsymbol{A}$.

The observed information matrix is usually more convenient to use than the expected information matrix, as it does not require an expectation to be taken. Thus, it is common in practice to estimate the inverse of the covariance matrix of a maximum-likelihood solution by the observed information matrix $\boldsymbol{I}(\hat{\boldsymbol{\Psi}};\boldsymbol{x})$, rather than the expected in-

formation matrix $\boldsymbol{\mathcal{I}}(\boldsymbol{\Psi})$ evaluated at $\boldsymbol{\Psi} = \hat{\boldsymbol{\Psi}}$. This approach gives the approximation

$$SE(\hat{\Psi}_i) \approx (\boldsymbol{I}^{-1}(\hat{\boldsymbol{\Psi}}; \boldsymbol{x}))_{ii}^{1/2}, \quad i = 1, \ldots, d. \tag{3.8}$$

Often the log likelihood function cannot be maximized analytically, that is, the likelihood equation has no explicit solutions, particularly in incomplete-data problems. In such cases, it may be possible to compute the MLE of $\boldsymbol{\Psi}$ iteratively. Next we will introduce some algorithms for calculating maximum likelihood estimates.

3.2 EM Algorithm

Incomplete data often result in complicated likelihood functions, where MLE's usually have to be computed iteratively. The Expectation-Maximization algorithm proposed by Dempster, Laird, and Rubin in a celebrated paper in 1977, popularly known as the EM algorithm, is a broadly applicable approach to the iterative computation of MLE's, useful in a variety of incomplete-data problems, where algorithms such as the Newton-type methods may turn out to be more complicated.

For many practical problems, the complete-data likelihood has a nice form, hence the EM algorithm exploits the reduced complexity of MLE given the complete data. The basic idea is to associate with the given

incomplete-data problem, a complete-data problem for which MLE is computationally more tractable; for instance, the complete-data problem chosen may yield a closed-form solution to the MLE or may be amenable to MLE computation with a standard computer package.

On each iteration of the EM algorithm, there are two steps—called the *expectation step* or the *E-step* and the *maximization step* or the *M-step*. Because of this, the algorithm is called the EM algorithm. The E-step consists in manufacturing data for the complete-data problem, using the observed data set of the incomplete-data problem and the current value of the parameters, so that the simpler M-step computation can be applied to this "completed" data set. More precisely, it is the log likelihood of the complete data problem that is "manufactured" in the E-step. As the log likelihood is based partly on unobservable data, it is replaced by its conditional expectation given the observed data, where this E-step is effected using the current fit for the unknown parameters. Starting from suitable initial parameter values, the E- and M-steps are repeated until convergence. Of course, the complete data problem is to be suitably chosen from the point of view of simplicity of the complete-data MLE's.

The likelihood functions for grouped data and conditional data have been given in Chapter 1; we can see that it is difficult to derive explicit

solutions directly, but complete data estimation is straightforward; we therefore employ the EM algorithm to compute the MLE's iteratively for the parameters of a mixture.

Nevertheless, the EM algorithm is not without its limitations, many of which came to light in attempting to apply it in certain complex incomplete-data problems, such as the conditional data problem; we deal with this problem via a Newton-type algorithm. When estimating the means and standard deviations we also use the Newton-type method in the M-step of the EM algorithm, since for some distributions even the likelihood for complete data has no explicit solutions to the means and standard deviations.

3.3 Newton-Type Methods

Newton-type methods are common computational methods for calculating the numerical solution of problems in optimization, which include the Newton-Raphson (NR) method, quasi-Newton methods, and modified Newton methods. We shall give a brief introduction to the NR method and quasi-Newton methods.

The Newton-Raphson method for solving the likelihood equation

$$\boldsymbol{S}(\boldsymbol{x}|\,\boldsymbol{\Psi}) = \mathbf{0}, \tag{3.9}$$

approximates the gradient vector $\boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi})$ of the log likelihood function $\log L(\boldsymbol{\Psi})$ by a linear Taylor series expansion about the current fit $\boldsymbol{\Psi}^{(s)}$ for $\boldsymbol{\Psi}$. This gives

$$\boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi}) \approx \boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi}^{(s)}) - \boldsymbol{I}(\boldsymbol{\Psi}^{(s)};\boldsymbol{x})(\boldsymbol{\Psi} - \boldsymbol{\Psi}^{(s)}). \tag{3.10}$$

A new fit $\boldsymbol{\Psi}^{(s+1)}$ is obtained by taking it to be a zero of the right-hand side of (3.10). Hence

$$\boldsymbol{\Psi}^{(s+1)} = \boldsymbol{\Psi}^{(s)} + \boldsymbol{I}^{-1}(\boldsymbol{\Psi}^{(s)};\boldsymbol{x})\, \boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi}^{(s)}). \tag{3.11}$$

If the log likelihood function is concave and unimodal, then the sequence of iterates $\{\boldsymbol{\Psi}^{(s)}\}$ converges to the MLE of $\boldsymbol{\Psi}$, in one step if the log likelihood is a quadratic function of $\boldsymbol{\Psi}$. When the log likelihood function is not concave, the Newton-Rahpson method is not guaranteed to converge from an arbitrary starting value. Under reasonable assumptions on $L(\boldsymbol{\Psi})$ and a sufficiently accurate starting value, the sequence of iterates $\boldsymbol{\Psi}^{(s)}$ produced by the Newton-Raphson method converges to a solution $\boldsymbol{\Psi}^*$ of (3.9) very fast, and this is regarded as the major strength of the Newton-Raphson method. But there can be potentially severe problems with this method in applications. Firstly, it requires at each iteration the computation of the $d \times d$ information matrix $\boldsymbol{I}(\boldsymbol{\Psi}^{(s)};\boldsymbol{x})$ (that is, the negative of the Hessian matrix), which is likely to become expensive very rapidly as d becomes large. One must allow for the storage

of the Hessian or some set of factors of it. Furthermore, the Newton-Raphson method in its basic form (3.11) requires for some problems an impractically accurate initial value for $\boldsymbol{\Psi}$ for the sequence of iterates $\{\boldsymbol{\Psi}^{(s)}\}$ to converge to a solution of (3.9).

Since the Newton-Raphson method requires the evaluation of $\boldsymbol{I}(\boldsymbol{\Psi}^{(s)}; \boldsymbol{x})$ on each iteration s, it immediately provides an estimate of the covariance matrix of its limiting value $\boldsymbol{\Psi}^*$ (assuming it is the MLE), through the inverse of the observed information matrix $\boldsymbol{I}^{-1}(\boldsymbol{\Psi}^*; \boldsymbol{x})$.

A broad class of methods are so-called quasi-Newton methods, for which the solution of (3.9) takes the form

$$\boldsymbol{\Psi}^{(s+1)} = \boldsymbol{\Psi}^{(s)} - \boldsymbol{A}^{-1}\boldsymbol{S}(\boldsymbol{x}|\boldsymbol{\Psi}^{(s)}), \tag{3.12}$$

where $\boldsymbol{A}$ is used as an approximation to the Hessian matrix. This approximation can be maintained by doing a update of $\boldsymbol{A}$ at each iteration. Methods of this class have the advantage over the Newton-Raphson method of not requiring the explicit evaluation of the Hessian matrix of the log likelihood function at every iteration.

Chapter 4

Rmix: A Package for Fitting Finite Mixture Distributions

4.1 Introduction

The objective of this project is to program statistical software for estimating the parameters in a mixture model, so we develop a package (i.e., library) called *Rmix* for the *R* statistical computing environment to fit finite mixture distributions, with the functionality of Macdonald's MIX software (1979), but with updated and substantially improved numerical methods based on a combination of the EM algorithm ([MK97]) and a Newton-type method implemented in the function `nlm` provided by the *R* system. A number of utility functions were also written to aid in preparing data and starting values for the parameters, plotting mixture distributions, providing the analysis of variance, and so on.

The evolution of the *R* system as a cross-platform open-code environment for statistical computing is one of the most exciting current developments in statistics. It is freely available world-wide and provides all the functions for graphics, statistical distributions, linear algebra, optimization, etc., that one commonly needs for statistical applications. This permits rapid development of elegant object-oriented code to implement statistically efficient algorithms. By publishing the results as an *R* package, this work will be immediately available to users world-wide on a variety of platforms.

4.2 The Functionality and Feature of *Rmix*

Rmix is an add-on package to the *R* system for data analysis and graphics. *Rmix* provides a suite of tools designed for statistical analysis of finite mixture distributions with grouped and conditional data. The components of the mixtures can be normal, log-normal, exponential or gamma, Weibull, binomial, negative binomial and Poisson distributions. The statistical method applied to fit the mixture is maximum likelihood estimation, and the algorithms used for the calculation of the MLE's are the EM algorithm and a Newton-type method.

In addition to the same functionality as Macdonald's MIX software, *Rmix* extends his work to fit Weibull, binomial, negative binomial and

Poisson distributions with some new constraints. *Rmix* can handle many mixture distribution applications, such as mixtures of exponential distributions for time-to-failure studies and scale mixtures with equal means for non-normal error analysis. Titterington *et al.* (1985) describe many applications of mixtures for which the current version of *Rmix* will give useful results.

In *Rmix* , there are three steps for fitting a mixture. *Rmix* assumes that the data are grouped, in the form of numbers of observations over successive intervals. Data often come grouped (as a histogram) or can be grouped with very little loss of information. Grouping greatly simplifies the calculation of maximum likelihood estimates ([MP79]). For *Rmix* , the grouping intervals are specified by their right-hand boundaries. The first (leftmost) and last (rightmost) intervals are open-ended; that is, if there are m intervals, the first interval includes everything up to the interval boundary a_1, the second everything from a_1 to a_2, and so on to the $m-1$st interval, which includes everything from a_{m-2} to a_{m-1}, and the mth, which includes everything above a_{m-1}. Thus it is only necessary to specify $m-1$ boundaries. The choice of boundaries is discussed in Macdonald and Pitcher (1979). As *Rmix* only deals with grouped data and conditional data, we first need to bin ungrouped data. *Rmix* also requires the data to be fitted and the initial values of parameters in the

specified format, and thus there are functions designed for grouping and formatting data.

Estimating the parameters of a mixture distribution is difficult when the components are heavily overlapped because the overlapping obscures information about individual components. The mixture can only be resolved by bringing additional information to the problem. The information could be from additional samples, or from some form of prior knowledge about the parameters or the relations among them. Thus, depending on the applications, we impose suitable constraints on the proportions, means and standard deviations to reduce the number of parameters. The constraints itemized in Chapter 2 are all allowed by *Rmix*, but some combinations of the constraints are not permitted (see section 2.4); a function for checking constraints is provided by *Rmix* .

In *Rmix* , we also provide a function that translates the whole collection of parameters occurring in a mixture model to the set of parameters to be estimated in terms of the constraints. For instance, in some applications it may be reasonable to assume that the coefficients of variation are all equal, i.e., CCV, and no constraints on the proportions and means. If we provide the function with the constraints (NONE for the proportions, NONE for the means and CCV for the sigmas), it will give the set of the parameters $(\pi_1, \ldots, \pi_{k-1}; \mu_1, \ldots, \mu_k; \sigma_1)$ to be estimated.

Users can start with as many constraints on the parameters as necessary and work repeatedly towards a solution that has as few constraints as possible and makes sense in terms of the application. All of the functions above are to prepare the elements for fitting mixture distributions.

Rmix allows users to incorporate additional data in the analysis, in the form of stratified sub-samples, such as conditional data: in length-frequency applications, sub-samples for age-determination would be taken at specific lengths, and analyzed jointly with the overall length-frequency distribution.

The main function `mix` is intended to realize the main functionality of *Rmix* , that is, to fit a mixture distribution to grouped and conditional data. The arguments users can specify include the data to be fitted, initial values of parameters, distribution of components, constraints on parameters, and so on. After running `mix`, an R object containing the estimated parameters, standard errors of the parameters estimated, chi-square statistic, P-value for the goodness-of-fit test, etc., is obtained.

The remaining functions are to do further analysis. The histogram of the current data can be displayed by the function `plot.mix`. The weighted component distributions $\pi_1 f(x|\mu_1, \sigma_1)$, ..., $\pi_k f(x|\mu_k, \sigma_k)$ and the mixture distribution $g(x|\,\boldsymbol{\Psi}) = \pi_1 f(x|\mu_1, \sigma_1) + \cdots + \pi_k f(x|\mu_k, \sigma_k)$ can be computed from the estimated parameter values and superimposed on

the histogram, and the positions of the means $\mu_1, \ldots, \mu_k$ on the x-axis are indicated with triangles. Users can judge whether the histogram is fitted well visually and then make some modifications to the initial values of parameters or to the distribution of components.

Rmix can also be used to test the goodness-of-fit of the model to the data and, in some cases, it can be used to test the validity of certain constraints. These tests depend on the chi-square approximation to the likelihood ratio statistic ([R65]) and will be valid as long as most of the intervals have expected counts of 5 or greater. The degrees of freedom are computed as the number of grouping intervals minus 1 minus the number of parameters estimated. After a successful fit, *Rmix* will compute a significance level (P-value) for the goodness-of-fit test.

If the data can be fitted with and without a certain constraint, the validity of that constraint can be tested. Removing the constraint will, in general, reduce the chi-square and the degrees of freedom; the reduction in chi-square is itself a chi-square statistic with degrees of freedom equal to the reduction in degrees of freedom ([R65], p. 350). This is only valid if the data give actual counts over intervals and if most counts are 5 or greater.

The goodness-of-fit test only indicates how well the mixture distribution $g(x)$ fits the histogram (or the grouped data) overall. If the

components overlap extensively the test is not very sensitive to features that are obscured by the overlapping, such as skewness of the component distributions.

Other functions can compute standard errors for all estimates, do the analysis of variance (ANOVA) for one or two fits, estimate joint, mixed or conditional data, and so forth.

4.3 Statistical and Numerical Methods

Next, we will illustrate the statistical methods and numerical algorithms applied in *Rmix* to calculate the MLE's for the parameters of a mixture distribution with grouped and conditional data.

4.3.1 Estimate Proportions for Fixed Means, Sigmas

Suppose that X is a random variable from a mixture distribution composed of 2-parameter components $\boldsymbol{\theta}_i = (\mu_i, \sigma_i)^T, i = 1, \ldots, k$. Then the p.d.f. of X can be written

$$g(x|\,\boldsymbol{\Psi}) = \sum_{i=1}^{k} \pi_i f(x|\,\mu_i, \sigma_i), \tag{4.1}$$

where $\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_{k-1}, \mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k)^T$. Let $x_1, \ldots, x_n$ be a sample of size n made on X. Instead of the raw data, we obtain the grouped data over m intervals with probabilities $P_j(\boldsymbol{\Psi})$, and n_j denotes

the frequency of the jth interval. The likelihood function for the grouped data has the multinomial form

$$L(\boldsymbol{\Psi}) = \frac{n!}{n_1! \cdots n_m!} \{P_1(\boldsymbol{\Psi})\}^{n_1} \cdots \{P_m(\boldsymbol{\Psi})\}^{n_m} . \tag{4.2}$$

Then the log likelihood is given by

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{m} n_j \log P_j(\boldsymbol{\Psi}) , \tag{4.3}$$

where

$$\begin{aligned} P_j(\boldsymbol{\Psi}) &= \int_{a_{j-1}}^{a_j} g(x|\, \boldsymbol{\Psi})\, dx \\ &= \int_{a_{j-1}}^{a_j} \left[\sum_{i=1}^{k} \pi_i f(x|\, \mu_i, \sigma_i) \right] dx \\ &= \sum_{i=1}^{k} \pi_i P_{j|i}(\boldsymbol{\theta}_i) . \end{aligned} \tag{4.4}$$

Substitution of (4.4) into (4.3) yields

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{m} n_j \log \left\{ \sum_{i=1}^{k} \pi_i P_{j|i}(\boldsymbol{\theta}_i) \right\}, \tag{4.5}$$

where

$$P_{j|i}(\boldsymbol{\theta}_i) = \int_{a_{j-1}}^{a_j} f(x|\, \mu_i, \sigma_i)\, dx , \quad j = 1, \ldots, m, \tag{4.6}$$

denotes the probability that an individual known to be from the ith group falls into the jth interval.

We first consider the estimation of proportions π_i in which the components of the mixture occur, where we assume the component densities

are completely specified; that is, the means and standard deviations of components are held fixed. On differentiating (4.5) with respect to π_i and equating the result to zero, we obtain

$$\sum_{j=1}^{m} n_j \left\{ \frac{P_{j|i}(\boldsymbol{\theta}_i)}{P_j(\boldsymbol{\Psi})} - \frac{P_{j|k}(\boldsymbol{\theta}_k)}{P_j(\boldsymbol{\Psi})} \right\} = 0\,, \quad i = 1, \ldots, k-1, \tag{4.7}$$

as the likelihood equation, which clearly does not yield an explicit solution for $\hat{\pi}_i$.

This problem can be solved within the EM framework by introducing the vectors

$$\boldsymbol{z}_j = (z_{1j}, \ldots, z_{kj})^T, \quad j = 1, \ldots, m,$$

as the missing data, where z_{ij} denotes the number of observations from the ith group falling into the jth interval. If these z_{ij} were observable, then the complete-data likelihood function for $\boldsymbol{\Psi}$ is equivalent to

$$L_c(\boldsymbol{\Psi}) = \prod_{i=1}^{k} \frac{z_{i+}!}{z_{i1}! \cdots z_{im}!} \{\pi_i P_{1|i}(\boldsymbol{\theta}_i)\}^{z_{i1}} \cdots \{\pi_i P_{m|i}(\boldsymbol{\theta}_i)\}^{z_{im}}\,, \tag{4.8}$$

with logarithm

$$\log L_c(\boldsymbol{\Psi}) = \sum_{i=1}^{k} \sum_{j=1}^{m} z_{ij} \log \pi_i + \sum_{i=1}^{k} \sum_{j=1}^{m} z_{ij} \log P_{j|i}(\boldsymbol{\theta}_i) + c_1, \tag{4.9}$$

where

$$c_1 = \sum_{i=1}^{k} \log \left\{ z_{i+}! \Big/ \prod_{j=1}^{m} z_{ij}! \right\}$$

does not depend on $\boldsymbol{\Psi}$. Then the MLE of π_i is simply given by

$$\hat{\pi}_i = \sum_{j=1}^{m} z_{ij}/n, \quad i = 1, \ldots, k, \tag{4.10}$$

which is the proportion of the sample having arisen from the ith component of the mixture.

If the constraint PFX is imposed on the proportions—that is, we assume that the proportions $\pi_{i1}, \pi_{i2}, \ldots, \pi_{it} (0 \leq t < k-1)$ are fixed—then the MLE's of the remaining proportions π_i are

$$\hat{\pi}_i = \frac{z_{i+}(1 - \pi_{i1} - \cdots - \pi_{it})}{n - z_{i1+} - \cdots - z_{it+}} \tag{4.11}$$

where

$$z_{i+} = \sum_{j=1}^{m} z_{ij}\,; \qquad z_{id+} = \sum_{j=1}^{m} z_{id,j}\,, \quad d = 1, \ldots, t.$$

We can see that (4.11) also holds when there are no constraints on the proportions.

As (4.9) is linear in the unobservable data z_{ij}, the E-step (on the $(s+1)$th iteration) simply requires the calculation of the current conditional expectation of Z_{ij} given the observed n_j, where Z_{ij} is the random variable corresponding to z_{ij}. Now

$$E^{(s)}(Z_{ij} | \, n_j, j = 1, \ldots, m) = z_{ij}^{(s)}\,, \tag{4.12}$$

where

$$z_{ij}^{(s)} = \tau_i(n_j | \, \boldsymbol{\Psi}^{(s)}) \tag{4.13}$$

and

$$\begin{aligned}\tau_i(n_j|\,\boldsymbol{\Psi}^{(s)}) &= n_j\, P_{i|j}(\boldsymbol{\theta}_i^{(s)}) \\ &= n_j\, \frac{\pi_i^{(s)} P_{j|i}(\boldsymbol{\theta}_i^{(s)})}{\sum_{i'=1}^{k} \pi_{i'}^{(s)} P_{j|i'}(\boldsymbol{\theta}_i^{(s)})}\end{aligned} \tag{4.14}$$

for $i = 1, \ldots, k; j = 1, \ldots, m$. The M-step on the $(s+1)$th iteration simply requires replacing each z_{ij} by $z_{ij}^{(s)}$ in (4.10) to give

$$\hat{\pi}_i^{(s+1)} = \frac{z_{i+}^{(s)}(1 - \pi_{i1} - \cdots - \pi_{it})}{n - z_{i1+}^{(s)} - \cdots - z_{it+}^{(s)}}, \tag{4.15}$$

for the proportions that are not fixed.

4.3.2 Estimate Means, Sigmas for Fixed Proportions

We now consider the estimations of the means and standard deviations for fixed proportions. For grouped data, the log likelihood function is given by (4.5). To derive the maximum likelihood estimate $\hat{\boldsymbol{\theta}}_i$, we differentiate (4.5) with respect to $\boldsymbol{\theta}_i$ and equate the result to zero; then we obtain the likelihood equation for $\boldsymbol{\theta}_i$ with the proportions fixed

$$\sum_{j=1}^{m} n_j\, \frac{\pi_i(\partial P_{j|i}(\boldsymbol{\theta}_i)/\partial \boldsymbol{\theta}_i)}{\sum_{i=1}^{k} \pi_i P_{j|i}(\boldsymbol{\theta}_i)} = 0 \quad i = 1, \ldots, k. \tag{4.16}$$

We can see that (4.16) has no solution with a closed form.

This problem can also be simplified within the EM framework. We use the same E-step as that for estimating the proportions, so that we

obtain the same log likelihood function as (4.9) for the complete data. As the proportions are fixed, this log likelihood can then be written

$$\log L_c(\boldsymbol{\theta}) = \sum_{i=1}^{k} \sum_{j=1}^{m} z_{ij} \log P_{j|i}(\boldsymbol{\theta}_i)\,. \tag{4.17}$$

There is no general form for the MLE's of the means and standard deviations since $P_{j|i}(\boldsymbol{\theta}_i)$ has different forms when different distributions and constraints are applied. For example, when the component densities are taken to be univariate normal with unknown means $\mu_1, \ldots, \mu_k$ and variances $\sigma_1^2, \ldots, \sigma_k^2$, and no constraints are imposed on the means and variances, the MLE's of μ_i and σ_i^2 are

$$\hat{\mu}_i = \sum_{j=1}^{m} z_{ij}\, x_j \Big/ \sum_{j=1}^{m} z_{ij}$$

and

$$\hat{\sigma}_i^2 = \sum_{j=1}^{m} z_{ij}(x_j - \mu_i)^2 \Big/ \sum_{j=1}^{m} z_{ij}\,.$$

If we assume that all the variances are equal, then we obtain the MLE of the common variance

$$\hat{\sigma}_i^2 = \sum_{i=1}^{k} \sum_{j=1}^{m} z_{ij}(x_j - \mu_i)^2/n\,.$$

However, for some distributions and constraints we can't derive the MLE's of the means and sigmas from the likelihood function (4.17) directly, and thus we use a Newton-type method in the M-step to compute the MLE's of the means and sigmas.

Differentiating

$$\mathfrak{L}_c(\boldsymbol{\theta}) = -2\sum_{i=1}^{k}\sum_{j=1}^{m} z_{ij}\, \log\left\{P_{j|i}(\boldsymbol{\theta}_i)/\hat{P}_{j|i}\right\}, \tag{4.18}$$

with respect to $\boldsymbol{\theta}_i$ and equating the result to zero, we can obtain the same equation as the likelihood equation of (4.17), so maximizing (4.17) is exactly equivalent to minimizing (4.18). Here $\hat{P}_{j|i} = z_{ij}/z_{i+}$ denotes the estimated relative frequency in the jth interval of the ith component, and the (i,j)th term in the sum is taken to be zero if $\hat{P}_{j|i} = 0$.

The reason why we work with the function (4.18) rather than (4.17) is that we want to make use of the function `nlm` provided by the *R* system, which carries out minimization of functions using a Newton-type algorithm, to compute the MLE's of the means and sigmas.

4.3.3 Estimate Proportions, Means, Sigmas with or without Constraints

We can also estimate proportions, means and sigmas at the same time by using the Newton-type method without the EM framework. To accomplish this purpose, we take advantage of the function `nlm` to compute the MLE's of all the parameters in a mixture distribution.

For grouped data, the log-likelihood function is

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{m} n_j \log P_j(\boldsymbol{\Psi}), \tag{4.19}$$

where $P_j(\boldsymbol{\Psi})$ denotes the theoretical probability that an individual belongs to the jth interval. Let $\hat{P}_j = n_j/n$ denote the observed relative frequency of the jth interval; then maximizing (4.19) is exactly equivalent to minimizing

$$\mathfrak{L}(\boldsymbol{\Psi}) = -2\sum_{j=1}^{m} n_j \, \log\left\{P_j(\boldsymbol{\Psi})/\hat{P}_j\right\}, \tag{4.20}$$

where the jth term in the sum is taken to be zero if $\hat{P}_j = 0$. Working with (4.20), which is twice the Kullback (1959) "minimum discrimination information statistic", is more convenient than working with (4.19) for three reasons: we can directly use the function `nlm` to carry out the minimization; (4.20) is the more clearly interpretable as a measure of discrepancy between $\hat{P}_j$ and $P_j(\boldsymbol{\Psi})$; and, in large samples, the minimized value of (4.20) can be used as a chi-squared statistic for testing the goodness-of-fit of the model.

For conditional data, we minimize the function

$$\begin{aligned}\mathfrak{L}_{condit}(\boldsymbol{\Psi}) \;&=\; -2\sum_{j=1}^{m} n_j \log\left\{\sum_{i=1}^{k} \pi_i P_{j|i}(\boldsymbol{\theta}_i)\right\} + \\ &\quad -2\sum_{jr,r=1,\ldots,h}\sum_{i=1}^{k} c_{i,jr}\log\left\{P_{i|j}(\boldsymbol{\Psi})/\hat{P}_{i|j}\right\}\end{aligned} \tag{4.21}$$

where

$$\hat{P}_{i|j} = c_{i,jr}\Big/\sum_{i=1}^{k} c_{i,jr}\,,$$

instead of maximizing the likelihood (1.9).

The Newton-type iterative method for minimizing (4.20) converges more quickly than the EM algorithm, but, in general, only when starting from an initial value very close to the minimum. An exception is the case where only the mixture proportions are being estimated. We therefore first run a few iterations of the EM algorithm to improve the initial values, and then use the Newton-type method to estimate the parameters and covariance matrix.

Chapter 5

Examples: Fit Mixture Distributions Using *Rmix*

5.1 Introduction

In this chapter we will illustrate the estimation procedure for fitting a mixture distribution using *Rmix*, and then give some examples and explain the results.

In this chapter, the `typewriter font` is used for *R* commands and functions, and *R* commands displayed are shown with the *R* prompt `>`. Commands that require more than one line of input are displayed with the *R* continuation prompt `+`.

5.2 Preparation for Fitting a Mixture

Start *Rmix* by starting *R*, and then loading the add-on package *Rmix* to the *R* session by using the following command:

```
> library(Rmix)
```

After attaching the *Rmix* functions, we shall begin preparing the data, the initial values of parameters and the constraints.

5.2.1 Organizing the Working Data

We have a data set consisting of the sample of $n = 523$ northern pike described in chapter 1. The mixture was known to comprise exactly five components. The five components correspond to the five age-groups present in the sample, all fish more than five years old having been removed from the sample. The age of each fish had been determined by scale-reading, and so this is a very suitable example for studying the statistical methodology.

The raw data called `pikraw` contain two columns, the first column displays the 523 lengths of pike, the second column gives the age-group of each fish:

```
> pikraw
```

```
     length  age
  1    18.0    1
  ⋮       ⋮    ⋮
 55    28.5    1
 56    25.0    2
  ⋮       ⋮    ⋮
298    41.5    2
299    31.0    3
  ⋮       ⋮    ⋮
454    53.0    3
455    44.5    4
  ⋮       ⋮    ⋮
501    64.5    4
502    49.5    5
  ⋮       ⋮    ⋮
523    76.5    5
```

As *Rmix* only deals with grouped data and conditional data, we have to reconstruct the raw data in the form of length-frequency data. To group raw data in the form described in section 2.4, one can use the function `mixgroup` as follows:

```
> pikgrp <- mixgroup(pikraw[,1],
+   breaks = c(0, seq(19.75, 65.75, 2), 80))
```

The grouped data `pikgrp` is an *R* object with class `"mixdata"` and `"data.frame"` which can be used for further calculation:

```
> pikgrp
    length  freq
  1  19.75     4
  2  21.75    10
  3  23.75    21
  4  25.75    11
  5  27.75    14
  6  29.75    31
  7  31.75    39
  8  33.75    70
  9  35.75    71
 10  37.75    44
 11  39.75    42
 12  41.75    36
 13  43.75    23
 14  45.75    22
 15  47.75    17
```

```
16    49.75    12
17    51.75    12
18    53.75    11
19    55.75     8
20    57.75     3
21    59.75     6
22    61.75     6
23    63.75     3
24    65.75     2
25      Inf     5
```

The intervals of grouped data should not be too wide or too narrow, or it is difficult to determine the means and standard deviations by inspecting the histogram. Also, most intervals should include at least 5 observations so that one can test the validity of a constraint.

If grouped data rather than raw data are given, one can enter the data directly from the keyboard to create a data frame by using the expression:

```
> pikgrp <- data.frame(length = c(19.75, 21.75, 23.75,
+   25.75, 27.75, 29.75, 31.75, 33.75, 35.75, 37.75,
+   39.75, 41.75, 43.75, 45.75, 47.75, 49.75, 51.75,
```

```
+   53.75, 55.75, 57.75, 59.75, 61.75, 63.75, 65.75, Inf),
+   freq = c(4, 10, 21, 11, 14, 31, 39, 70, 71, 44, 42,
+   36, 23, 22, 17, 12, 12, 11, 8, 3, 6, 6, 3, 2, 5))
> class(pikgrp) <- c("mixdata", "data.frame")
```

Data files also can be created beforehand using a text editor. Write the name of the measurement, `"length"` here, a space and `"freq"` on the first line. Then write the pairs of right boundaries and frequencies, starting each pair on a new line and separating the right boundary from the frequency by a space. End with `"Inf"` and the frequency from the rightmost interval, again on a new line. Place an empty line at the end of the file. Make sure that the boundaries are in strictly ascending order. Then data can be read from the file in table format by the command:

```
> pikgrp <- read.table("d:/temp/pikdat.txt", header = T)
> class(pikgrp) <- c("mixdata", "data.frame")
```

The data frames obtained by the three methods are identical.

The raw data also contain the age of each fish; we can display them by using the expression:

```
> pikcondit <- mixgroup(pikraw, breaks =
+   c(0, seq(19.75, 65.75, 2), 80), usecondit = T)
```

Users also can add conditional data to the grouped data using the function:

```
> pikcondit <- conditdat(pikgrp, 5, c(c(4, 9, 2, 0, 0, 0),
+   c(5, 8, 6, 0, 0,0), c(12, 0, 2, 34, 0, 0),
+   c(13, 0, 0, 21, 0, 0), c(15, 0, 0, 5, 5, 0),
+   c(16, 0, 0, 6, 5, 1), c(17, 0, 0, 5, 7, 0),
+   c(18, 0, 0, 4, 4, 3), c(19, 0, 0, 0, 8, 0),
+   c(20, 0, 0, 0, 2, 1), c(21, 0, 0, 0, 1, 5),
+   c(22, 0, 0, 0, 2, 4)))

> pikcondit
    length  freq  C1  C2  C3  C4  C5
 1   19.75     4   0   0   0   0   0
 2   21.75    10   0   0   0   0   0
 3   23.75    21   0   0   0   0   0
 4   25.75    11   9   2   0   0   0
 5   27.75    14   8   6   0   0   0
 6   29.75    31   0   0   0   0   0
 7   31.75    39   0   0   0   0   0
 8   33.75    70   0   0   0   0   0
 9   35.75    71   0   0   0   0   0
10   37.75    44   0   0   0   0   0
```

```
11   39.75   42   0   0   0   0   0
12   41.75   36   0   2  34   0   0
13   43.75   23   0   0  21   0   0
14   45.75   22   0   0   0   0   0
15   47.75   17   0   0   5   5   0
16   49.75   12   0   0   6   5   1
17   51.75   12   0   0   5   7   0
18   53.75   11   0   0   4   4   3
19   55.75    8   0   0   0   8   0
20   57.75    3   0   0   0   2   1
21   59.75    6   0   0   0   1   5
22   61.75    6   0   0   0   2   4
23   63.75    3   0   0   0   0   0
24   65.75    2   0   0   0   0   0
25     Inf    5   0   0   0   0   0
```

Conditional data from a prepared file or from the keyboard can also be used to produce the same data frame by applying similar expressions to those for grouped data.

5.2.2 Determining Starting Values for Parameters

For fitting a mixture using iterative algorithms, a complete set of initial parameter values is required, and the format is specified in *Rmix*. The starting values should be in the form of a data frame containing three variables, which are, in order, the proportions π_i, means μ_i, and standard deviations σ_i. None of the starting values for the three variables can be missing (NA or NaN) or infinity (Inf), and the proportions can only have values between 0 and 1. Besides, the sigmas cannot be negative or zero. The components must be indexed so that the means are in non-decreasing order, $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_k$. If any two consecutive means are equal the corresponding standard deviations must be in strictly ascending order. That is, $\mu_i = \mu_{i+1}$ is allowed only if $\sigma_i < \sigma_{i+1}$. *Rmix* will not accept starting values unless these requirements are satisfied.

In a typical application the parameters of the component distribution would not be known initially: only the histogram would be available. However, after some experience with graphs of this type one can do a reasonable job of sketching in the component distributions if the number of components is known and if something about the population being studied is available, for example, they are known to be about equally spaced and about equal in standard deviation. Note, however, that this would not be possible in the present example if the number of

components were unknown. The components do not all form distinct modes and there is therefore no indication from the histogram alone as to the number of components to expect. While estimating the number of components from the data alone is an attractive possibility, estimation will clearly be unreliable in cases such as this and we will not consider the problem in this report.

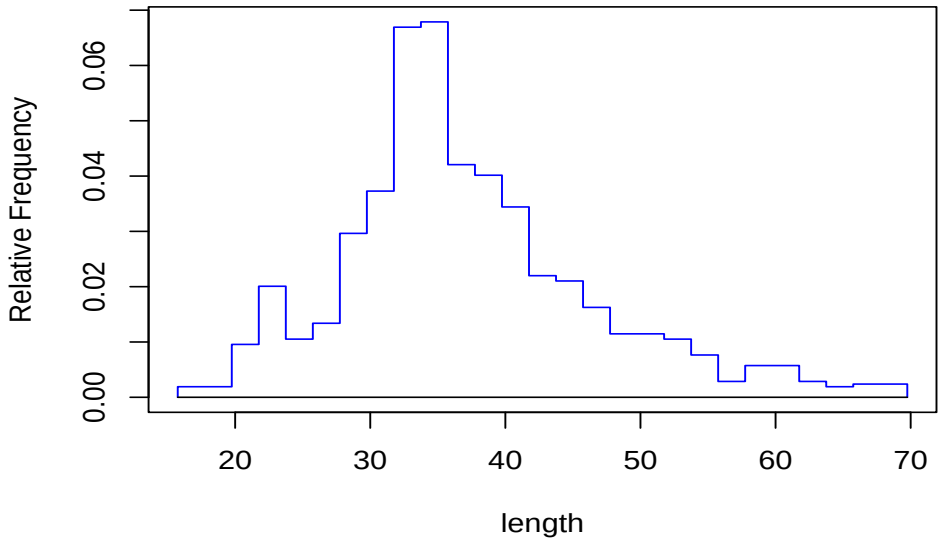
Starting values for the means and sigmas should be as good as possible, since the iterations will not converge if the starting values and constraints are not well chosen. Starting values for the proportions are less critical, since they can usually be improved very efficiently. Users can find starting values for the means and sigmas by inspecting the histogram, and *Rmix* provides a function for plotting the histogram with or without the weighted components distributions and mixture distribution superimposed on it.

In Figure 5.1 the length-frequency histogram is displayed by using the method function `plot.mixdata` called via the generic function `plot`:

```
> plot(pikgrp)
```

Although the leftmost (first) and rightmost (mth) intervals are always open-ended (section 4.2), on the histogram the first interval is shown as being twice the width of the second and the mth is shown as being twice the width of the $(m-1)$st.

Figure 5.1: Pike length-frequency histogram



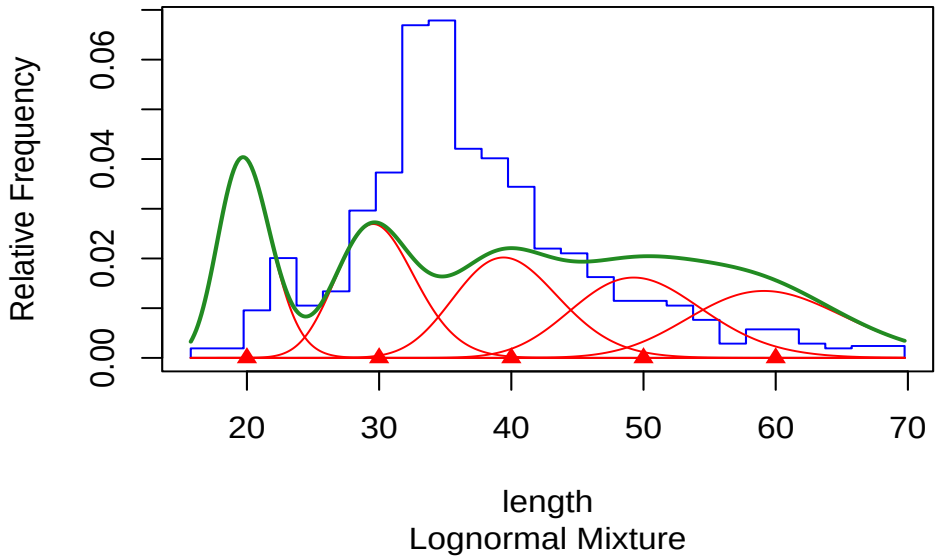
In this example, we determine the starting values for the means and sigmas visually, and we just make the proportions equal. A simple way to enter the initial values is to use the function `mixparam` as follows:

```
> pikpar <- mixparam(c(20, 30, 40, 50, 60),
+   c(2, 3, 4, 5, 6), rep(0.2, 5))

> pikpar
    pi  mu  sigma
 1  0.2  20     2
 2  0.2  30     3
 3  0.2  40     4
 4  0.2  50     5
 5  0.2  60     6
```

Then we plot the histogram (Figure 5.2) with the components using the starting values, to check if the means and sigmas are reasonable. Lognormal distributions are assumed for the components.

```
> plot(pikgrp, pikpar, "lnorm")
```

Figure 5.2: Pike lengths histogram with initial parameter values



To help determine and modify the starting values, one can regard the thick curve in Figure 5.2 as providing a fit to the sample histogram. Then try to envisage how the "goodness-of-fit" is affected by slight alterations in the parameters, particularly the means and standard deviations. Altering a mean will shift the corresponding component to the left or right; increasing a standard deviation will cause the corresponding component to spread and flatten, while decreasing a standard deviation will produce

a narrower and higher peak. It is always better to choose initial values for the standard deviations that are too small, rather than too large. Large standard deviations cause the components to overlap more than is necessary, obscuring the resolution of the means. Changing either the mean or the variance a little would make the fit much better or worse and so we conclude that the fit is sensitive to these parameters. The important corollary to this is that fitting the model to the data will yield good estimates of these parameters. Applying these observations to Figure 5.2, we can see that the first and second means can be improved. Nevertheless, one reason why the mixture (the thick curve) doesn't fit the histogram well is that the proportions have not yet been fitted.

Furthermore, the starting values should be consistent with the constraints and the distribution of components. For example, if one wants to constrain the means to lie along a growth curve, then $(\mu_3 - \mu_2) < (\mu_2 - \mu_1)$ is required. Also, negative means are not permitted by the constraints `"FCV"`, `"CCV"`, `"BINOM"`, `"NBINOM"`, `"POIS"` and all the distributions but normal. If binomial distribution components with the constraint `"BINOM"` are fitted, then the initial values need to satisfy $\mu_i > \sigma_i^2$. And negative binomial components with the constraint `"NBINOM"` require $\mu_i < \sigma_i^2$.

5.2.3 Preparing the Constraints

Choosing the constraints mainly depends on some form of prior information about the parameters or the relations among them. The histogram also provides some information. Two examples are given in section 1.1 of Macdonald ([MG88]).

Further, there are intimate relations among the parameters, the constraints and the distribution of components. So the selection of constraints is restricted by the relations (see section 2.4). Users can use the function `mixconstr` to organize the constraints in the form *Rmix* requires and to check if the constraints meet those requirements.

5.3 The Estimation Procedure for Fitting Mixtures

After preparing the data, starting values and constraints, we can begin to estimate the parameters of a mixture. The function `mix` performs the whole estimation procedure for fitting a mixture.

5.3.1 Fit Mixtures with Grouped Data

For the grouped data `pikgrp`, we try to fit the lognormal distribution to all 5 age-groups with constant coefficient of variation using the combination of the EM algorithm and Newton-type method:

```
> fitccv <- mix(mixdat = pikgrp, mixpar = pikpar,
+   dist = "lnorm", constr = mixconstr(consigma = "CCV"),
+   emsteps = 10)
```

If the EM procedure is employed, the number of EM steps needs to be provided. Usually, 10 or 20 will be more than adequate. If the value is 0, *Rmix* will skip this procedure without changing any of the parameter values. In our example, the number of EM steps is 10. The parameter values obtained by the EM estimation procedure are displayed as follows:

```
           pi       mu    sigma
1 0.09427418 22.84942 2.227289
2 0.35195367 32.37254 3.155572
3 0.33950791 37.77252 3.681944
4 0.15493256 47.51442 4.631554
5 0.05933167 59.66492 5.815946
```

We can see that the proportions are much changed from the starting values after the 10 EM steps. The final estimates of the parameters are listed below:

```
> fitccv
           pi       mu    sigma
1 0.09967030 23.07345 2.372187
```

```
2  0.51889286  33.60686  3.455129
3  0.22676635  41.10281  4.225790
4  0.10710022  49.88250  5.128431
5  0.04757028  60.46696  6.216622
Distribution:
"lnorm"
Constraints:
 conpi   conmu  consigma
"NONE"  "NONE"     "CCV"
```

And the estimates of parameters from complete data are given by:

```
      pi     mu  sigma
1  0.105  23.33   2.44
2  0.465  33.09   3.00
3  0.298  41.27   4.27
4  0.090  51.24   5.08
5  0.042  61.32   7.07
```

The two sets are very similar.

5.3.2 Fit Mixtures with Conditional Data

We fit a mixture to the conditional data `pikcondit` using the command:

```
> fitcondit <- mix(mixdat = pikcondit, mixpar = pikpar,
+   dist = "lnorm", constr = mixconstr(consigma = "CCV"),
+   usecondit = T)
```

There are no EM steps for conditional data. The estimates of the parameters are:

```
> fitcondit$parameters
           pi        mu    sigma
1  0.10228190  23.17540  2.238091
2  0.48298761  33.16981  3.203271
3  0.29571900  41.65567  4.022766
4  0.07501706  51.85026  5.007277
5  0.04399443  60.83656  5.875100
```

Compared with the estimated parameter values for the grouped data, we can see that the estimates obtained from the conditional data are more accurate, since additional information about the population is provided by the subsamples.

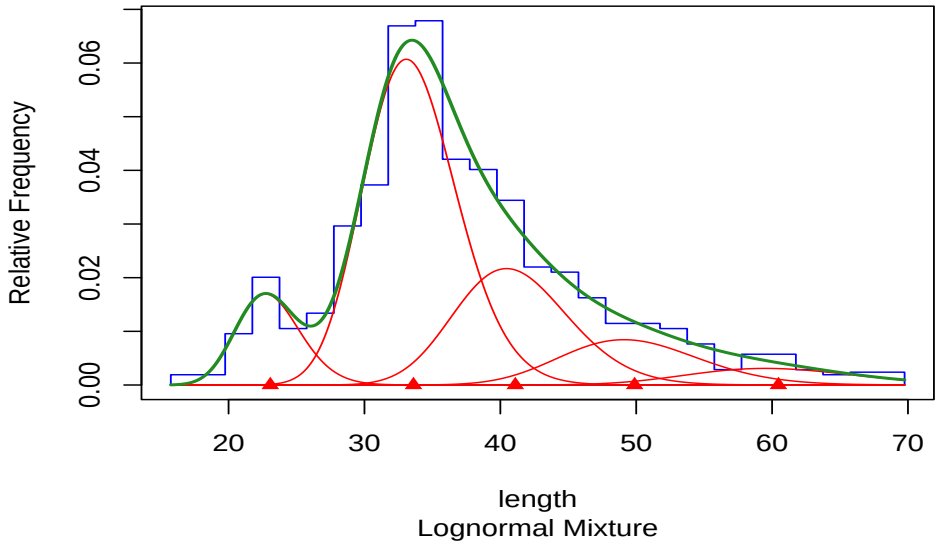
5.4 Other Issues

After fitting the mixture to the data, it is desirable to see intuitively how well we have done, and thus we plot the histogram for the grouped

data with the estimated values of parameters (Figure 5.3). It can be seen that the result obtained by *Rmix* is quite a reasonable fit.

```
> plot(fitccv)
```

Figure 5.3: Pike lengths histogram with estimated parameters



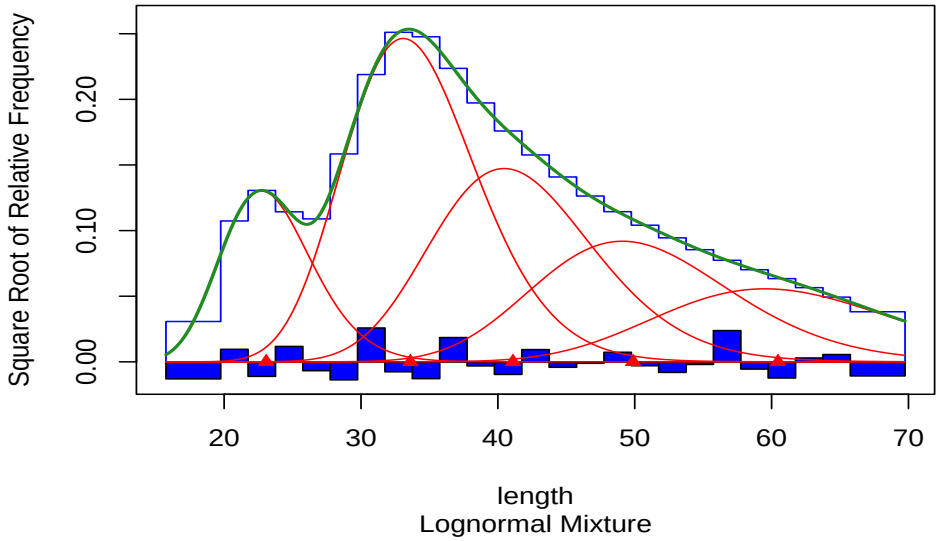
We can also draw a hanging rootogram (Figure 5.4) to visualize departures from the model.

```
> plot(fitccv, root = T)
```

As the rectangles around the x-axis are small and no areas are with the rectangles only above or below the x-axis, we can say that the model fits the histogram well.

Next, we attempt to test the goodness-of-fit of the model. *Rmix* provides the method function `anova.mix` to accomplish this purpose, called via the generic function `anova`:

Figure 5.4: Pike lengths rootogram with fitted curves




```
> anova(fitccv)

 Analysis of Variance Table
           Df   Chisq  Pr(>Chisq)
Residuals  14  11.948     0.6105
```

The P-value is 0.6105, which indicates the model estimated is consistent with the data.

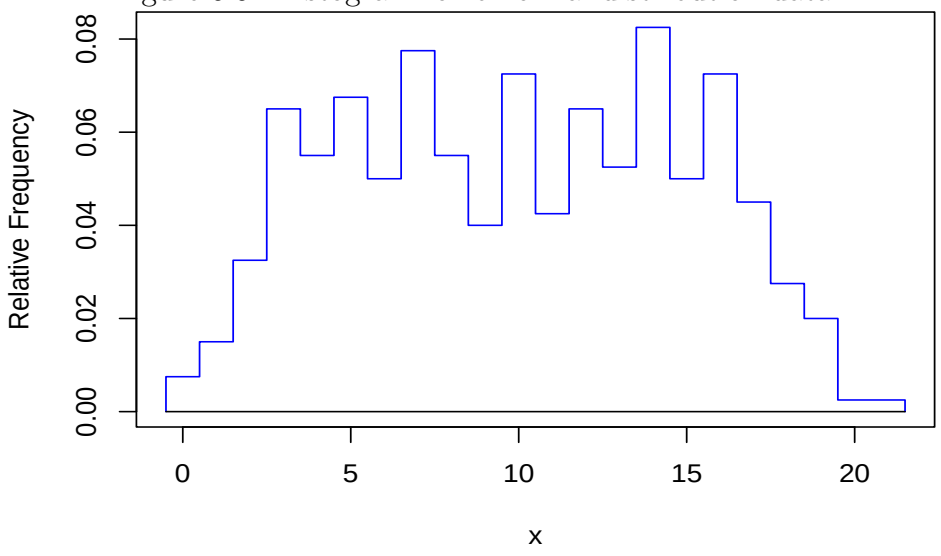
We can also use this function to test the validity of certain constraints. For example, if we fit the mixture with not only constant coefficient of variation but also means that lie along a curve, then the model is "nested" within the model with constant coefficient of variation. From the ANOVA table given by the function `anova.mix`, one can test the validity of the constraint `"MGC"` (see section 4.2).

Furthermore, *Rmix* has a function to give the estimated data such as $z_{ij}, i = 1, \ldots, k; j = 1, \ldots, m$. Please refer to the documentation for the function `fitted.mix` in the Appendix for more information.

5.5 Examples

We randomly generate four groups of binomial distribution data with means 4, 8, 12, 16, and corresponding variances 3.2, 4.8, 4.8 and 3.2. Then we mix the four data groups with 100 observations for each group, i.e., with equal proportions. After grouping the mixture data, we plot the histogram (Figure 5.5) for the grouped data.

Figure 5.5: Histogram for binomial distribution data



We give starting values for parameters as follows:

```
> binpar
     pi  mu  sigma
1  0.25   3    1.5
2  0.25   7    2.0
3  0.25  13    2.0
4  0.25  15    1.5
```

We use the function `mix` to fit the binomial mixture.

```
> fitbin <- mix(mixdat = bindat, mixpar = binpar,
+   dist = "binom", constr = mixconstr(consigma = "BINOM",
+   size = c(20, 20, 20, 20)))
```

The estimated values obtained are

```
> fitbin
       pi      mu  sigma  size
1  0.2322   3.702  1.737    20
2  0.3046   7.895  2.186    20
3  0.3221  13.365  2.106    20
4  0.1410  16.768  1.646    20
Distribution:
"binom"
Constraints:
```

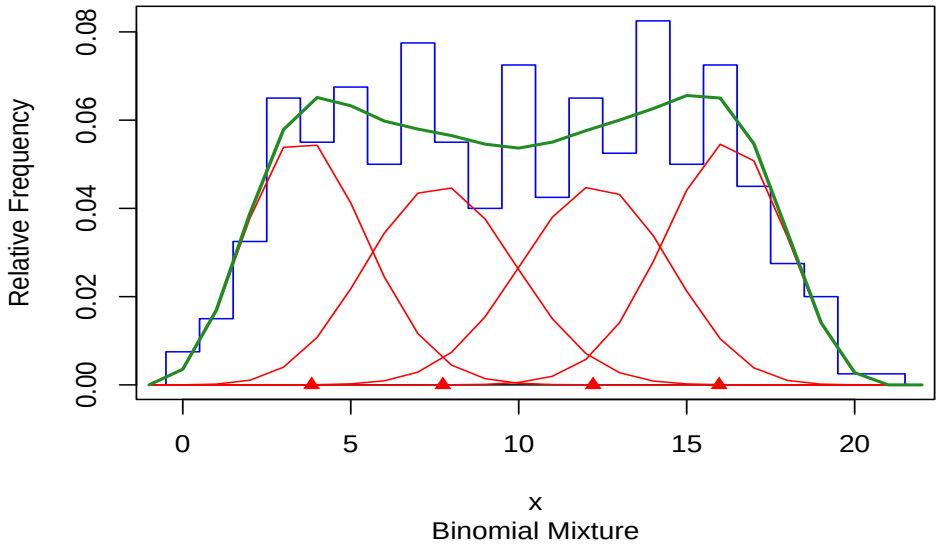
```
   conpi   conmu  consigma
  "NONE"  "NONE"   "BINOM"
```

We can see that the estimated means and standard deviations are close to the real values, but the proportions not so close. Next we fix all the proportions at the starting values, and the results are much better, that is, they are very close to the true values. Figure 5.6 gives the histogram with fitted curves.

```
> fitbinpfx <- mix(mixdat = bindat, mixpar = binpar,
+   dist = "binom", constr = mixconstr(conpi = "PFX",
+   fixpi = c(T, T, T, T), consigma = "BINOM",
+   size = c(20, 20, 20, 20)))

> fitbinpfx
     pi      mu  sigma  size  fixpi
 1  0.25   3.836  1.761    20      T
 2  0.25   7.743  2.178    20      T
 3  0.25  12.215  2.181    20      T
 4  0.25  15.965  1.795    20      T

 Distribution:
 "binom"

 Constraints:
```

```
conpi   conmu  consigma

"PFX"  "NONE"   "BINOM"
```

Figure 5.6: Histogram with fitted curves for binomial distribution data



Chapter 6

Discussion and Suggestions for Future Research

6.1 Comparison of the EM Algorithm and Newton-type Methods

We employ two numerical methods to fit a mixture in our project, which are presented in the last sections. Here we shall give some comparative remarks:

1. The EM algorithm is usually simple to apply and satisfies the appealing monotonic property.

2. The EM algorithm is less sensitive to starting values.

3. Newton-type methods are more complicated, particularly in view of the matrix inversion required, and there is no guarantee of mono-

tonicity.

4. If the Newton-type methods converge, convergence is of second order (i.e. fast), whereas the EM algorithm is often excruciatingly slow.

5. The iterations for Newton-type methods incorporate approximations to the covariance matrix of $\hat{\boldsymbol{\Psi}}$, the maximum likelihood estimator of $\boldsymbol{\Psi}$. This is not available directly with the EM algorithm.

6. Unconditional convergence to $\hat{\boldsymbol{\Psi}}$ is not guaranteed with any of the techniques. This is not surprising and is a familiar characteristic of Newton-type methods. Of course, even with the monotonic EM algorithm we would not necessarily obtain convergence to a global maximum.

6.2 Strategies for Difficult Cases

Difficult cases arise under some specific conditions, such as when iterations diverge, some of the estimated proportions are close to zero, and so on. Thus, some strategies will be provided for avoiding the occurrence of those situations.

6.2.1 What to do when Iterations will not Converge

We have introduced the algorithms employed in *Rmix*, and discussed their respective strengths and weakness. We can see that convergence sometimes cannot be guaranteed for Newton-type algorithms, especially when the starting values and constraints are not well chosen.

For determining starting values, the histogram of data is definitely helpful. The initial guesses from inspection of the histogram can be revised by superimposing the component density curves with the starting values on the original histogram to see how well they fit. Any attempt to fit without constraints on the parameters, in particular on the means and sigmas, usually results in divergence since too many parameters are estimated simultaneously. A strategy is to impose enough constraints first to prevent the iterations from diverging, and then to repeat the fitting process with fewer constraints to see how much the goodness-of-fit and the estimates depend on that choice.

In the procedure of estimation, we suggest that users run 1, 2 or 3 iterations of the EM algorithm. There are three reasons for this recommendation: one is that the proportions and the means, sigmas are estimated separately; that is, the proportions are first estimated with fixed means and sigmas, and then holding the proportions fixed, means and sigmas are computed (see section 4.3). Thus the number of param-

eters being estimated at one time will be reduced. The second reason is that the EM algorithm bears the monotonic property, so that it may improve the initial values to prevent the iterations for the Newton-type method from diverging. The last reason is that the estimated values of parameters can be printed at each step if the argument `'print.level'` takes the value `'3'`, so that the user can know how the estimation process is going.

A warning message indicating the reason why the optimization process terminated will be given when the estimation procedure is completed without convergence. The message often provides important information. For example, if a warning saying "the optimization process terminated because iteration limit exceeded" is given, then the user may try to give a larger number for the limit of iterations performed to see if the iterations will converge.

6.2.2 What to do when Proportions Go Close to 0

Rmix may, in some cases, produce some proportions close to 0, and at the same time the corresponding components will disappear on the histogram.

If this happens, it probably indicates that too many components are fitted. The user may reset the initial values of parameters with fewer

components. It is also possible that there really is a component there, but the current value of its mean places it too far into one of the neighboring components. There are then several strategies to choose from: plot the current fit and see if any components are obviously misplaced; try other distributions or constraints; set the offending proportion to a small positive value and hold it fixed at that value.

6.2.3 What to do when Parameters Have Large Standard Errors

It sometimes turns out that one or more of the parameters have exceedingly large standard errors associated with them, or the standard errors cannot be computed because the information matrix is singular (Macdonald and Pitcher 1979). This will happen if there is no information in the data for one or more of the parameters, or if the current parameter values are very far from their true values. In either case, inspection of the plot with the current parameter values and consideration of what the solution ought to be, should suggest a revision of the starting values or constrains that will be more successful on the next attempt.

6.3 Suggestions for Future Research

Three suggestions for future research are as follows:

1. For the iteration scheme of Newton-type methods, the calculation of the gradient and Hessian of the likelihood function, i.e. the first and second partial derivatives with respect to each parameter being estimated, is necessary. Thus, whichever of the likelihood functions (4.19) and (1.9) is employed, we have to differentiate $P_j(\boldsymbol{\Psi})$ with respect to each element of $\boldsymbol{\Psi}$. However, the forms of the gradient and Hessian will change when different component distributions and constraints are applied; therefore all of the forms need to be considered. Because of the scope of this project, we only give the likelihood functions, but don't write routines to compute the gradient and Hessian; instead, we make use of the function `nlm` provided by the *R* computing statistical software to carry out the minimization of the functions (4.18), (4.20) and (4.21). The algorithm employed by `nlm` is a Newton-type method. Although the documentation for `nlm` provides plenty of information, it's still not enough to understand clearly each step of the minimization procedure, and so the general method for computing various forms of derivatives and routines to achieve the functionality of `nlm` should be investigated in future research.

2. For any software, it is possible that some functionality desirable for specific users is not provided. *Rmix* would not be an excep-

tion; nevertheless, the power of the *R* environment is that one can combine elements in new ways, the result of a calculation is an object available for further calculation, and the user can easily define various functions to achieve specified purposes, so one can enhance the capability of *Rmix* by adding functions to it. We shall give an example to illustrate how a user can extend *Rmix*. Suppose that a user is trying to force the means to lie along a growth curve, but this curve is different from the von Bertalanffy growth curve defined by *Rmix*, so the constraint MGC cannot be used directly in this specific situation. However, the user can attempt to program a function, we assume, named `vbgc` to compute points on this specific curve and add the function to *Rmix*. When the user attempts to fit a mixture with means fixed and on the growth curve, he can just call the function `vbgc()` inside `mixparam()`. Therefore, the routines that can enhance the functionality of *Rmix* should be explored further.

3. The EM algorithm bears the property of monotonicity, but it usually converges very slowly, while the much more rapid Newton-type methods have no guarantee to converge. So modifications of the algorithms to overcome the limitations should be considered for future versions of *Rmix*.

Appendix

1. **anova.mix**

```
ANOVA Tables for Mixture Model Objects

Description:

    Compute analysis of variance tables for one or two
    mixture model objects.

Usage:

    anova.mix(mixobj1, mixobj2)

Arguments:

 mixobj1: an object of class '"mix"', usually, the results
          returned by the model fitting function 'mix'.

 mixobj2: an object of the same type to be compared with
          'mixobj1', which contains the results of fitting
          another model with more or fewer parameters
          fitted.
```

Value:

An object of class ‘anova’ inheriting from class ‘data.frame’. When given a single argument this function produces a table which tests whether the model is significant. The table contains the residual degrees of freedom, Chi-square statistic and P value. If the class of the argument is not ‘"mix"’, it returns ‘NULL’. When given two objects, it tests the models against one another and lists them in the order of number of parameters fitted. For the model with fewer parameters fitted, the change in degrees of freedom is given. This only make statistical sense if the models are nested. If one of arguments does not belong to the class ‘"mix"’, the function will give the anova table for the other argument; if both of them do not, it returns ‘NULL’.

Warning:

The comparison between two models will only be valid if they are fitted to the same data set. And the two models should be nested.

```
See also:

    The model fitting function ‘mix’, the generic function
    ‘anova’.

Examples:

    data(pikdat)   # load the grouped data ‘pikdat’
    data(pikpar0)   # load the parameters ‘pikpar0’
    anova.mix(pikdat)   # NULL with a warning message
    fitpik1 <- mix(pikdat, pikpar0, "lnorm", mixconstr(
                conmu = "MFX", fixmu = c(F, F, F, F, T),
                consigma = "CCV"), emstep = 10)
    anova.mix(fitpik1)
    fitpik2 <- mix(pikdat, pikpar0, "lnorm", mixconstr(
                consigma = "CCV"), emsteps = 10)
    anova.mix(fitpik1, fitpik2)
    anova.mix(fitpik2, fitpik1)
```

2. **coef.mix**

```
Extract Mixture Model Coefficients

Description:

    ‘coef.mix’ is a function which extracts mixture model
```

coefficients from objects returned by the model fitting function ‘mix’. It is called via the generic function ’coef’.

Usage:

```
coef(mixobj, natpar = F)
```

Arguments:

mixobj: an object of class ‘"mix"’, usually, the results returned by the model fitting function ‘mix’.

natpar: a logical scalar specifying whether the natural parameters should be given.

Value:

A data frame containing three variables, which are, in order, the proportions, means, and standard deviations, respectively. If ‘natpar’ is ‘TRUE’, then the natural parameters of component distributions are also displayed.

See also:

‘mix’ for model fitting.

Examples:

```
    data(pikdat)   # load the grouped data 'pikdat'
    data(pikpar0)   # load the data set 'pikpar0'
    fit <- mix(pikdat, pikpar0, "lnorm", mixconstr(
              consigma = "CCV"), emsteps = 10)
    coef(fit)
    coef(fit, natpar = T)
```

3. **conditdat**

Add Conditional Data to Grouped Data

Description:

It adds conditional data to grouped data.

Usage:

```
conditdat(mixdat, k, conditsamples)
```

Arguments:

```
 mixdat: a data frame containing grouped data, whose
         first column should be the right boundaries of
         grouping intervals, and the second one should
         be the numbers of observations falling into
         each interval.
      k: the number of components.
```

```
 conditsamples: a vector containing conditional data, which
                consists of the conditional samples, the
                first element of each sample is the number
                indicating which interval this sample comes
                from.

Value:

    A data frame containing the grouped data with
    conditional data.

See also:

    'mixgroup' for constructing grouped and conditional
    data.

Examples:

    data(pikdat)   # load the data set 'pikdat'
    pikdat   # display the data set 'pikdat'
    conditdat(pikdat, k = 5, conditsamples =
           c(c(4, 9, 2, 0, 0, 0), c(5, 8, 6, 0, 0,0),
             c(12, 0, 2, 34, 0, 0), c(13, 0, 0, 21, 0, 0),
             c(15, 0, 0, 5, 5, 0), c(16, 0, 0, 6, 5, 1),
             c(17, 0, 0, 5, 7, 0), c(18, 0, 0, 4, 4, 3),
             c(19, 0, 0, 0, 8, 0), c(20, 0, 0, 0, 2, 1),
```

```
            c(21, 0, 0, 0, 1, 5), c(22, 0, 0, 0, 2, 4))))
      # add conditional data to the grouped data 'pikdat'
```

4. **covmat**

Compute Standard Errors of the Estimated Parameters

Description:

Compute the covariance matrix and standard errors of the estimated parameters of a mixture distribution.

Usage:

```
covmat(mixpar, constr, hessian)
```

Arguments:

mixpar:	A data frame containing the parameter values of component distributions, which are, in order, the proportions, means, and standard deviations.
constr:	a list of constraints on parameters of component distributions.
hessian:	the hessian at the estimated maximum of log likelihood function.

Value:

A list containing the following items:

vmat: the covariance matrix of the estimated parameters of component distributions.

mixse: a data frame giving the standard errors of estimated parameters.

See also:

'mix' for fitting mixture distributions, 'mixconstr' for constructing constraints.

5. **fitted.mix**

Compute Mixture Model Fitted Values

Description:

'fitted.mix' is a function which computes fitted values from objects returned by modeling function 'mix'. It is called via the generic function 'fitted'.

Usage:

```
fitted(mixobj, digits = NULL)
```

Arguments:

mixobj: an object of class ‘"mix"’, usually, the results returned by the model fitting function ‘mix’.

digits: a specified number of decimal places to be reserved.

Value:

List with the following components:

mixed: the estimated mixed data, that is, the fitted numbers of observations falling into each interval.

joint: the estimated joint data, that is, the fitted numbers of observations from each component falling into every interval.

conditional: the estimated conditional data to be returned if ‘usecondit’ of ‘mixobj’ is ‘TRUE’, which are the fitted numbers of observations from given intervals belonging to each component.

conditprob: the estimated conditional probabilities of observations from given interval belonging to each component.

```
See also:

    'mix' for fitting mixture distributions.

Examples:

    fit1 <- mix(pikdat, pikpar0, "lnorm", mixconstr(
              consigma = "CCV"), emsteps = 10)
    fitted(fit1)
    fit2 <- mix(pike65sg, pikpar0, "gamma", mixconstr(
              consigma = "CCV"), usecondit = T)
    fitted(fit2, digits = 2)
```

6. **groupstats**

```
Estimate Parameters of One-Component Mixture Distribution

Description:

    'groupstats' is a function which estimates the
    proportion, mean and standard deviation for a mixture
    distribution with one component.

Usage:

    groupstats(mixdat)

Arguments:
```

mixdat: A data frame containing grouped data, whose first column should be right boundaries of grouping intervals where the first and last intervals are open-ended; whose second column should consist of the frequencies indicating numbers of observations falling into each interval.

Value:

List with the following components:

pi: the value is '1' because of only one component.

mean: the estimated mean of this component.

sigma: the estimated standard deviation of this component.

See also:

'mixparam' for constructing starting values of parameters.

Examples:

```
data(pikdat)
groupstats(pikdat)
```

7. grintprob

Compute Probabilities of an Observation Falling into a

Grouping Interval

Description:

Compute probabilities of an observation falling into a grouping interval when given component distribution the observation comes from.

Usage:

```
grpintprob(mixdat, mixpar, dist, constr)
```

Arguments:

mixdat:	A data frame containing grouped data, whose first column should be right boundaries of grouping intervals where the first and last intervals are open-ended; whose second column should consist of the frequencies indicating numbers of observations falling into each interval.
mixpar:	A data frame containing the parameter values of component distributions, which are, in order, the proportions, means, and standard deviations.
dist:	the distribution of components, it can be ‘"norm"’, ‘"lnorm"’, ‘"gamma"’, ‘"weibull"’, ‘"binom"’, ‘"nbinom"’ and ‘"pois"’.

```
 constr: a list of constraints on parameters of component
         distributions.
Value:
    It produces a matrix, whose each column contains the
    probabilities of the observations from one component
    falling into each grouping interval.
Examples:
    data(bindat)
    data(binpar)
    grpintprob(bindat, binpar, "binom", mixconstr(
              consigma = "BIN", size = c(20, 20, 20, 20)))
```

8. **mix**

```
Estimate Parameters of Mixture Distributions
Description:
    Find a set of overlapping component distributions that
    gives the best fit to grouped data and conditional
    data, using a combination of EM algorithm and
    Newton-type methods.
```

Usage:

```
mix(mixdat, mixpar, dist = "norm", constr = list(
    conpi = "NONE", conmu = "NONE", consigma = "NONE",
    fixpi = NULL, fixmu = NULL, fixsigma = NULL,
    cov = NULL, size = NULL), emsteps = 1,
    usecondit = F, exptol = 5e-06, print.level = 0, ...)
```

Arguments:

mixdat: A data frame containing grouped data, whose first column should be right boundaries of grouping intervals where the first and last intervals are open-ended; whose second column should consist of the frequencies indicating numbers of observations falling into each interval. If conditional data are available, this data frame should have $k+2$ columns, where k is the number of components, whose element in row j and column $i+2$ is the number of observations from the jth interval belonging to the ith component.

mixpar: A data frame containing starting values for parameters of component distributions, which are, in order, the proportions, means, and standard

deviations.

dist: the distribution of components, it can be ‘"norm"’, ‘"lnorm"’, ‘"gamma"’, ‘"weibull"’, ‘"binom"’, ‘"nbinom"’ and ‘"pois"’.

constr: a list of constraints on parameters of component distributions. See function ‘mixconstr’.

emsteps: a non-negative integer specifying the number of EM steps to be performed.

usecondit: if ‘usecondit’ is ‘TRUE’ and ‘mixdat’ includes conditional data, then conditional data will be used with grouped data to estimate parameters of mixtures.

exptol: a positive scalar giving the tolerance at which the scaled fitted value is considered large enough to be a degree of freedom.

print.level: this argument determines the level of printing which is done during the optimization process. The default value of ‘0’ means that no printing occurs, a value of ‘1’ means that initial and final

```
          details are printed and a value of '2' means
          that full tracing information is printed.
      ...: additional arguments to the optimization function
          'nlm'.
Value:
    A list containing the following items:
   parameters: A data frame containing estimated values for
               parameters of component distributions, which
               are, in order, the proportions, means, and
               standard deviations.
           se: A data frame containing estimated values for
               standard errors of parameters of component
               distributions.
 distribution: the distribution used to fit the data.
   constraint: the constraints on parameters.
        chisq: the goodness-of-fit chi-square statistic.
           df: degrees of freedom of the fitted mixture
               model.
            P: a significance level (P-value) for the
               goodness-of-fit test.
```

```
      vmat: covariance matrix for the estimated
            parameters.
   mixdata: the original data, i.e.  the argument ‘mixdat’.
 usecondit: the value of the argument ‘usecondit’.
```

See also:

‘mixgroup’ for grouping data, ‘mixparam’ for organizing the parameter values, ‘mixconstr’ for constructing constraints. ‘nlm’ for additional arguments.

Examples:

```
data(pikdat)
data(pikpar0)
mix(pikdat, pikpar0, "lnorm", constr = mixconstr(
    consigma = "CCV"), emsteps = 10)
data(bindat)
data(binpar)
mix(bindat, binpar, "binom", constr = mixconstr(
    consigma = "BINOM", size = c(20, 20, 20, 20)))
```

9. **mixconstr**

Construct Constraints on Parameters

Description:

Construct constraints on parameters and check if the constraints are invalid. See the reference for details.

Usage:

```
mixconstr(conpi = "NONE", conmu = "NONE", consigma =
          "NONE", fixpi = NULL, fixmu = NULL,
          fixsigma = NULL, cov = NULL, size = NULL)
```

Arguments:

conpi: a constraint on proportions, it can be either ‘"NONE"’ denoting no constraint on proportions, or ‘"PFX"’ indicating some proportions being fixed.

conmu: a constraint on means, it can be ‘"NONE"’, ‘"MFX"’, ‘"MEQ"’, ‘"MES"’ and ‘"MGC"’, which denote no constraint, specified means fixed, means equal, means with equal spaces and means lying along a growth curve, respectively.

consigma: a constraint on standard deviations, it can be ‘"NONE"’, ‘"SFX"’, ‘"SEQ"’, ‘"FCV"’, ‘"CCV"’, ‘"BINOM"’, ‘"NBINOM"’ and ‘"POIS"’, which denote no constraint, specified standard deviations fixed, standard deviations equal, fixed coefficient of variation, constant coefficient of variation, the means and standard deviations have the same relation as that of Binomial distribution, as that of Negative Binomial distribution and as that of Possion distribution.

fixpi: ‘NULL’ or a vector with ‘TRUE’ and ‘FALSE’ as its elements, indicating which proportions are fixed when ‘conpi’ is ‘"PFX"’. If an element is ‘TRUE’, the corresponding proportion is fixed at the starting value.

fixmu: similar to ‘fixpi’. ‘NULL’ or a vector indicating which means are fixed when ‘conmu’ is ‘"MFX"’.

fixsigma: similar to ‘fixpi’. ‘NULL’ or a vector indicating which standard deviations are fixed

when ‘consigma’ is ‘"MFX"’.

cov: ‘NULL’ or a scalar if ‘consigma’ is ‘"FCV"’, then the coefficients of variation are fixed at this scalar.

size: ‘NULL’ or a vector of numbers of trials for each component when ‘consigma’ is ‘"BINOM"’ or ‘"NBINOM"’.

Value:

A list containing the following components, which are, in order, ‘conpi’, ‘conmu’, ‘consigma’, ‘fixpi’, ‘fixmu’, ‘consigma’, ‘cov’, ‘size’.

References:

Macdonald, P.D.M. and Green, P.E.J. (1988) *User's Guide to Program MIX: An Interactive Program for Fitting Mixtures of Distributions.* ICHTHUS DATA SYSTEMS.

See also:

‘mixgroup’ for grouping data, ‘mixparam’ for constructing starting values of parameters.

Examples:

```
mixconstr()
mixconstr(conmu = "MEQ", consigma = "SFX", fixsigma =
          c(T, F, T, T, F))
mixconstr(consigma = "BINOM", size = c(25, 25, 25))
```

10. **mixgroup**

Construct Grouped Data from Raw Data

Description:

Group raw data in the form of numbers of observations over successive intervals.

Usage:

```
mixgroup(x, breaks = NULL, xname = NULL, k = NULL,
         usecondit = F)
```

Arguments:

x: a data frame or matrix containing raw data, whose first column should be the measurements to be grouped, and the second one, if available, includes the numbers indicating which component each individual belongs to.

```
    breaks: one of:
            1: a vector giving the boundaries of intervals
               which raw data are grouped into,
            2: a single number giving the number of
               intervals,
            3: a character string naming an algorithm to
               compute the number of intervals,
            4: a function to compute the number of
               intervals.
           In the last three cases the number is a
           suggestion only.
     xname: the name of measurement.
         k: the number of components.
 usecondit: if 'usecondit' is 'TRUE' and 'x' has two
           columns, then conditional data will be
           displayed with grouped data.
Value:
    A data frame containing grouped data derived from raw
    data, whose first column includes the right boundaries
    of grouping intervals, where the first and last
```

intervals are open-ended; whose second column consists of the frequencies which are the numbers of observations falling into each interval. If 'usecondit' is 'TRUE' and the numbers indicating which component the individual comes from are available, conditional data which can be regarded as a table, whose element in row j and column i is the number of observations from the jth interval belonging to the ith component, will be displayed with grouped data.

See also:

'hist' for more information about the argument 'breaks', 'mixparam' for organizing the parameter values, 'mixconstr' for constructing constraints.

Examples:

```
data(pikraw)   # load the data set 'pikraw'
pikraw   # display the data set 'pikraw'
mixgroup(pikraw)   # group raw data
mixgroup(pikraw, usecondit = T, k = 3)   # construct
   grouped data associated with conditional data
mixgroup(pikraw, usecondit = T, k = 8)
```

```
    mixgroup(pikraw, breaks =
             c(0, seq(19.75, 65.75, 2), 80),
             usecondit = T, k = 5)
```

11. **mixpar2theta**

Find the Parameters to be Estimated

Description:

When there are constraints on parameters, we only estimate some parameters in terms of the constraints. This function is to find the parameters to be estimated. See the reference for details.

Usage:

```
mixpar2theta(mixpar, constr, mixprop = T)
```

Arguments:

mixpar:	A data frame containing the values for parameters of component distributions, which are, in order, the proportions, means, and standard deviations.
constr:	a list of constraints on parameters of component distributions.

mixprop: if 'TRUE', the proportions will be estimated.

Value:

a vector containing the values for the parameters to be estimated.

References:

Macdonald, P.D.M. and Green, P.E.J. (1988) *User's Guide to Program MIX: An Interactive Program for Fitting Mixtures of Distributions.* ICHTHUS DATA SYSTEMS.

See also:

'mix' for fitting mixture model, 'mixtheta2par' for computing all the parameters from the estimated parameters.

Examples:

```
mixpar2theta(pikpar, constr = mixconstr(consigma =
          "CCV"))
mixpar2theta(pikpar, constr = mixconstr(conmu = "MGC",
          consigma = "SFX", fixsigma = c(F, F, F, T,
          T)), mixprop = F)
```

12. **mixparam**

Construct Starting Values for Parameters

Description:

Construct starting values for parameters of a mixture model.

Usage:

```
mixparam(mu, sigma, pi = NULL)
```

Arguments:

mu: a vector of means of component distributions, which should be in ascending order.

sigma: a vector of standard deviations of component distributions, which are corresponding to the means. 'sigmas' must be in ascending order when means are equal.

pi: the corresponding mixing proportions of components. If 'NULL', the proportions will be taken as $1/k$, where k is the length of 'mu'.

Value:

A data frame containing three variables, which are,

in order, the proportions, means, and standard deviations, respectively.

See also:

'mixgroup' for grouping data, 'mixconstr' for constructing constraints.

Examples:

```
mixparam(mu = c(20, 30, 40), sigma = c(2, 3, 4))
mixparam(c(20, 30, 40), c(3), c(0.15, 0.78, 0.07))
```

13. **mixtheta2par**

Compute All of Parameters from the Estimated Parameters

Description:

When there are constraints on parameters, we only estimate some parameters in terms of the constraints. This function is to compute all of parameters from the estimated ones.

Usage:

```
mixtheta2par(mixtheta, mixpar, constr, mixprop = T)
```

```
Arguments:
 mixtheta: a vector containing the values for the estimated
           parameters, usually, a result of the function
           'mixpar2theta'.
    mixpar: A data frame containing the values for
           parameters of component distributions, which
           are, in order, the proportions, means, and
           standard deviations.
    constr: a list of constraints on parameters of component
           distributions.  See function 'mixconstr'.
  mixprop: if 'TRUE', the proportions are estimated.
Value:
    A data frame containing three variables, which are,
    in order, the proportions, means, and standard
    deviations, respectively.
See also:
    'mix' for fitting mixture model, 'mixpar2theta' for
    finding the parameters to be estimated.
Examples:
    mixtheta2par(mixtheta = c(30, 2, 3, 4, 5, 6), pikpar,
                 mixconstr(consigma = "MEQ"), mixprop = F)
```

```
    par <- mixpar2theta(pikpar, constr = mixconstr(
                        consigma = "CCV"))
    mixtheta2par(par, pikpar, mixconstr(consigma = "CCV"))
```

14. **plot.mix**

Mix Object Plotting

Description:

A function for plotting of Mix objects. It is called via the generic function 'plot'.

Usage:

```
plot(mixobj, mixpar = NULL, dist = "norm", root = F,
        ytop = NULL, clwd = 1, main, sub, xlab, ylab,
        bty, ...)
```

Arguments:

mixobj: an object of class '"mix"', usually, the results returned by the model fitting function 'mix'.

mixpar: 'NULL' or a data frame containing the values for parameters of component distributions, which are, in order, the proportions, means, and standard deviations.

```
 dist: the distribution of components, it can be
       '"norm"', '"lnorm"', '"gamma"', '"weibull"',
       '"binom"', '"nbinom"' and '"pois"'.
 root: if 'TRUE', a hanging rootogram will be displayed.
 ytop: a specified value for the top of the y axis.
 clwd: a positive number denoting the line width,
       defaulting to '1'.
 main: an overall title for the plot.
  sub: a subtitle for the plot.
 xlab: a title for the x axis.
 ylab: a title for the y axis.
  bty: A character string which determined the type of box
       which is drawn about plots.  If 'bty' is one of
       '"o"', '"l"', '"7"', '"c"', '"u"', or '"]"' the
       resulting box resembles the corresponding upper
       case letter.  A value of '"n"' suppresses the box.
  ...: additional arguments to the function 'plot.default'.
Details:
    If the argument 'mixobj' gives an object of class
    '"mix"', the plot will be a histogram for the grouped
```

data come from the element ‘mixdata’ of ‘mixobj’. Although the leftmost (first) and rightmost (mth) intervals are always open-ended, on the histogram the first interval is shown as being twice the width of the second and the mth is shown as being twice the width of the $m-1$st. When the fitted distribution is one of ‘"lnorm"’, ‘"gamma"’ and ‘"weibull"’, the left boundary of the first interval will be taken zero since negative values and zeroes are not allowed for these distribution. For the distributions ‘"binom"’, ‘"nbinom"’ and ‘"pois"’ negative data are not permitted, so the left boundary of the first interval is taken -0.5. The component distributions weighted by their respect proportions and the mixture distribution are computed by the estimated parameter values from the element ‘parameters’ of ‘mixobj’, and superimposed on the histogram. The distribution of components will be taken the value of the element ‘distribution’. If ‘sub’, ‘xlab’, ‘ylab’ and ‘bty’ are not specified, the default values will be used. The positions of the means are indicated with

```
    triangles.

    When the argument 'root' is 'TRUE', a hanging
    rootogram will be displayed, that is, if there is only
    the grouped data, this option plots the histogram
    with the square root of relative frequency on the
    y-axis.  If there is a model as well as data, not only
    is the y-axis the square root of relative frequency,
    also the bars of the histogram, instead of rising from
    0, are shifted up or down so that the mid-point of the
    top of the bar is exactly on the curve indicating the
    mixture distribution and the bottom of the bar may
    therefore be above or below the x-axis.  If the bar
    goes below the x-axis, the portion below is shown as a
    blue rectangle.  If the bar does not reach the x-axis,
    the space between the bottom of the bar and the
    x-axis is shown as a blue rectangle.  If the blue
    rectangles are almost above or below in an area of
    the x-axis, we may say that the mixture curve around
    that area is not fitting well.

See also:

    'mixparam' for organizing the parameter values, 'mix'
```

```
    for fitting mixture model, ‘plot.default’ for
    additional arguments.

Examples:

    fit1 <- mix(pikdat, pikpar0, "lnorm", constr =
                mixconstr(consigma = "CCV"), emsteps = 10)
    plot(fit1)
    plot(fit1, root = T)
    fit2 <- mix(bindat, binpar, "binom", constr =
                mixconstr(consigma = "BINOM", size =
                c(20, 20, 20, 20)))
    plot(fit2)
    plot(fit2, root = T)
```

15. **plot.mixdata**

```
Mixdata Object Plotting

Description:

    A function for plotting of Mixdata objects.  It is
    called via the generic function ’plot’.

Usage:
```

```
    plot(mixdata, mixpar = NULL, dist = "norm", root = F,
         ytop = NULL, clwd = 1, main, sub, xlab, ylab,
         bty, ...)

Arguments:

 mixobj: an object of class `"mixdata"', usually, the
         results returned by the model fitting function
         `mixgroup'.

  Other arguments are the same as those of the function
  'plot.mix'.

Details:

    When the argument `mixdata' is a data frame containing
    grouped data, the histogram of the data will be
    displayed if `mixpar' is `NULL'; if it gives the
    parameters, the component distributions and the
    mixture are computed from the parameter values and
    superimposed on the histogram.

See also:

    `plot.mix' for plotting the mix object, `plot.default'
    for additional arguments.

Examples:
```

```
    plot(pikdat)

    plot(pikdat, pikpar0, "lnorm")

    plot(pikdat, pikpar0, "lnorm", root = T)
```

16. **print.mix**

```
Print Objects of Class ‘"Mix"’

Description:

    ‘print.mix’ is a function which prints objects of
    class ‘"mix"’ and returns it invisibly.  It is called
    via the generic function ’print’.

Usage:

    print(mixobj, digits = 4)

Arguments:

 mixobj: an object of class ‘"mix"’, usually, the results
         returned by the model fitting function ‘mix’.

 digits: how many significant digits are to be used.

Details:

    This function only prints information about the
    mixture model, which are the estimated parameters of
```

```
    the mixture, the distribution of components and the
    constraints on the parameters.  Also, the values for
    the parameters are rounded to the specified number
    of decimal places (default 4).  The whole object can
    be printed out using the function ‘print.default’.

See also:

    ‘mix’ for model fitting.  ‘print.default’ for
    printing the whole object.

Examples:

    fit <- mix(pikdat, pikpar0, "lnorm", mixconstr(
               consigma = "CCV"), emsteps = 10)
    fit
    print(fit)
    print.default(fit)
```

17. **summary.mix**

```
Summarizing Mixture Model Fits

Description:

    ‘summary’ method for class ‘"mix"’.  It is called via
    the generic function ’summary’.
```

Usage:

```
summary(mixobj, digits = 4)
```

Arguments:

mixobj:	an object of class ‘"mix"’, usually, the results returned by the model fitting function ‘mix’.
digits:	how many significant digits are to be used.

Value:

A list containing the following items:

parameters:	A data frame containing the values for parameters of component distributions, which are, in order, the proportions, means, and standard deviations.
standard errors:	a data frame giving the standard errors of estimated parameters.
anova table:	analysis of variance table for the ‘mixobj’, that is, the results from the function ‘anova.mix’.

See also:

‘mix’ for fitting mixture distributions, ‘summary’ for summarizing other kinds of object. ‘anova.mix’ for information about ‘anova table’.

Examples:

```
fit <- mix(pikdat, pikpar0, "lnorm", mixconstr(
            consigma = "CCV"), emsteps = 10)
summary(fit)
```

18. **testconstr**

Check Constraints

Description:

Check if constraints on parameters are valid.

Usage:

```
testconstr(mixdat, mixpar, dist, constr)
```

Arguments:

mixdat: A data frame containing grouped data, whose first column should be right boundaries of grouping intervals, whose second column should consist of the frequencies indicating numbers of observations falling into each interval. If conditional data are available, this data frame should have $k+2$ columns, where k is the number of components,

whose element in row j and column $i+2$ is the number of observations from the jth interval belonging to the ith component.

mixpar: A data frame containing the values for parameters of component distributions, which are, in order, the proportions, means, and standard deviations.

dist: the distribution of components, it can be ‘"norm"’, ‘"lnorm"’, ‘"gamma"’, ‘"weibull"’, ‘"binom"’, ‘"nbinom"’ and ‘"pois"’.

constr: a list of constraints on parameters of component distributions. See function ‘mixconstr’.

Value:

If the constraints are valid, this function will give a logical value ‘TRUE’. If not, it gives an error message to illustrate the reason.

See also:

‘mixgroup’ for grouping data, ‘mixparam’ for organizing the parameter values, ‘mixconstr’ for constructing constraints, ‘testpar’ for checking parameters.

Examples:

```
    testconstr(pikdat, pikpar, "lnorm", constr =
               mixconstr(consigma = "CCV"))
    testconstr(bindat, binpar, "binom", constr =
               mixconstr())
    testconstr(bindat, binpar, "binom", constr =
               mixconstr(consigma = "BINOM"))
    testconstr(bindat, binpar, "pois", constr =
               mixconstr(conmu = "MEQ", consigma =
               "POIS"))
```

19. **testpar**

```
Check Parameters

Description:

    Check if the values of parameters are valid.

Usage:

    testpar(mixpar, dist, constr)

Arguments:

 mixpar: A data frame containing the values for parameters
         of component distributions, which are, in order,
         the proportions, means, and standard deviations.
```

dist: the distribution of components, it can be ‘"norm"’, ‘"lnorm"’, ‘"gamma"’, ‘"weibull"’, ‘"binom"’, ‘"nbinom"’ and ‘"pois"’.

constr: a list of constraints on parameters of component distributions. See function ‘mixconstr’.

Value:

logical. If ‘TRUE’, the parameters are valid. If ‘FALSE’, some of the parameters are invalid. Since this function is for internal use, it doesn’t give error messages.

Details:

Any of the parameter values can not be missing value (‘NA’ or ‘NaN’) or infinity (‘Inf’), and the proportions can only take the values between 0 and 1. Besides, the standard deviations can not be negative. The components must be indexed so that the means are in non-decreasing order. If any two consecutive means are equal the corresponding standard deviations must be in strictly ascending order. Furthermore, the parameter values should be consistent with the constraints and the distribution

of components. For example, if one wants to constraint the means to lie along a growth curve, then $(\mu_3 - \mu_2) < (\mu_2 - \mu_1)$ is required. Also, negative means are not permitted by the constraints '"FCV"', '"CCV"', '"BINOM"', '"NBINOM"', '"POIS"' and all the distributions but Normal. If the Binomial distribution components with the constraint '"BINOM"' are fitted, then the relation $\mu_i > \sigma_i^2$ need to be satisfied. And the Negative Binomial components with the constraint '"NBINOM"' require $\mu_i < \sigma_i^2$.

See also:

'mixparam' for organizing the parameter values, 'mixconstr' for constructing constraints, 'testconstr' for checking constraints.

20. **weibullpar**

Compute Shape and Scale Parameters for Weibull Distribution

Description:

Compute the shape and scale parameters for weibull

```
    distribution given the mean, standard deviation and
    location.

Usage:

    weibullpar(mu, sigma, loc = 0)

Arguments:

    mu: the mean of weibull distribution.

 sigma: the standard deviation of weibull distribution.

   loc: the location parameter of weibull distribution.

Value:

    A data frame containing three variables, which are, in
    order, the shape, scale, and location parameter.

See also:

    'weibullparinv' for computing the mean and standard
    deviation from the shape, scale and location
    parameters.

Examples:

    weibullpar(2, 1.2)
    weibullpar(2, 1.2, 1)
```

21. **weibullparinv**

Compute the Mean and Standard Deviation of Weibull Distribution

Description:

Compute the mean and standard deviation given the values of shape, scale and location of weibull distribution.

Usage:

```
weibullparinv(shape, scale, loc = 0)
```

Arguments:

shape:	the shape parameter of weibull distribution.
scale:	the scale parameter of weibull distribution.
loc:	the location parameter of weibull distribution defaulting to 0.

Value:

A data frame containing three variables, which are, in order, the mean, standard deviation and location parameter.

See also:

```
    'weibullpar' for computing the shape and scale
    parameters from the mean and standard deviation.

Examples:

    weibullpar(2, 1.2, 1)
    weibullparinv(shape = 0.837612, scale = 0.910627,
                  loc = 1)
```

Bibliography

[DLR77] Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). *Maximum likelihood from incomplete data via the EM algorithm.* J.R. Statist. Soc., **B39**, 1-38.

[DS83] Dennis, J.E. and Schnabel, R.B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ.

[MP79] Macdonald, P.D.M. and Pitcher, T.J. (1979). *Age-groups from size-frequency data: a versatile and efficient method of analysing distribution mixtures.* Journal of the Fisheries Research Board of Canada, **36**, 987-1001.

[MG88] Macdonald, P.D.M. and Green, P.E.J. (1988). *User's Guide to Program MIX: An Interactive Program for Fitting Mixtures of Distributions, Release 2.3.* ICHTHUS DATA SYSTEMS.

[M92] Mangal, Milton E. (1992). *Constrained estimation of mixture normal distributions by the EM algorithm.* McMaster University Press.

[MB88] McLachlan, Geoffrey J. and Basford, Kaye E. (1988). *Mixture models.* Marcel Dekker, Inc.

[MJ90] McLachlan, G.J., and Jones, P.N. (1990). *Algorithm AS 254: Maximum likelihood estimation from grouped and truncated Data with finite normal mixture model.* Appl. Statist., **39, No.2**, 273-312.

[MK97] McLachlan, Geoffrey J., and Krishnan, Thriyambakam (1997). *The EM Algorithm and Extensions.* JOHN WILEY & SONS, INC.

[M92] Milton Eric Mangal, Grad. Dip. (1992). *Constrained Estimation of Mixture Normal Distributions by the EM algorithm* McMaster University.

[R65] Rao, C.R. (1965). *Linear statistical inference and its applications.* Wiley, New York.

[SKW85] Schnabel, R.B., Koontz, J.E. and Weiss, B.E. (1985). *A modular system of algorithms for unconstrained minimization.* ACM Trans. Math. Software, 11, 419-440.

[TSM85] Titterington, D.M., Smith, A.F.M., and Makov, U.E. (1985). *Statistical analysis of finite mixture distributions.* John Wiley & Sons.

[WRE02] *Writing R Extensions, Version 1.5.0.* R Development Core Team.